TREBALL FINAL DE GRAU

TFG TITLE: Enroute vortex conflict detection system for RPAS operations

DEGREE: Grau en Enginyeria d'aeronavegació

AUTHOR: Mònica Marcos Benítez

ADVISOR: Marc Pérez Batlle

DATE: July 18, 2016

Títol: Sistema de detecció de conflictes amb vortex en ruta per a operacions de RPAS

Autor: Mònica Marcos Benítez

Director: Marc Pérez Batlle

Data: 18 de juliol de 2016

Resum

En aquest projecte s'estudia l'afectació del vòrtex generat per una aeronau comercial sobre un RPAS (Remotely Piloted Aircraft Systems) quan ambdós es troben en ruta. L'objectiu és realitzar un sistema de detecció de vòrtex que permeti saber si el RPAS pot realitzar sense preocupació la seva ruta assignada, o si l'haurà de modificar com a conseqüència de l'existència del vòrtex generat per l'aeronau comercial, produint una situació de risc, on el RPAS podria bolcar i per tant tenir un accident o produir-ne un altre. Per fer-ho, es realitzarà un model de vòrtex, el qual serà utilitzat en el model de col·lisió. A més a més, haurà dos models de col·lisió. Un considerant que l'aeronau comercial va en línia recta, i un altre tenint en compte que l'avió pot dur a terme girs per canviar de rumb.

Title: Enroute vortex conflict detection system for RPAS operations

Author: Mònica Marcos Benítez

Advisor: Marc Pérez Batlle

Date: July 18, 2016

Overview

In this project the vortex affectation generated by a commercial airliner to a RPAS (Remotely Piloted Aircraft Systems) when both are at enroute phase is studied. The objective is to perform a vortex detection system that allows the RPAS to know if it can fly through its assigned route with no risks, or if it will have to modify it as a consequence of the vortex existence generated by the commercial airliner, producing a hazard situation, where the RPAS could turned upside down and therefore, to have an accident or produce another one. To execute it, a vortex model will be performed, which will be used at the collision model. Furthermore, there will be two collision models. One will consider the airliner flying in a straight line, and the other one will take into account that the commercial aircraft can make turns in order to change its heading.

TABLE OF CONTENTS

LIST OF FIGURES

Introduction

In few years, the airspace has experienced an increase in the air traffic density and what is expected is that it will grow more in the near future. In addition, an increased disparity in the size and weight of aircraft flying at cruise altitudes is also expected with the introduction of the RPAS in civil airspace flying together for instance with the super heavy aircraft, the A380. A part from its physical differences, it is well known that the RPAS have a different aircraft performance (regarding climb/descent performance or cruise airspeed) and they will probably perform flight profiles different from typical airliners (i.e. they will loiter over certain areas performing surveillance rather than point to point *missions*).

The introduction of the RPAS in civil airspace, which are susceptible to wake turbulence, along with super heavy or heavy aircraft, which are powerful generators, new potential wake turbulence conflicts which have not been considered before may appear. Wake vortex encounters may not only result into hazard situations near airports, but can also cause a significant threat in the upper airspace.

For this reason, the objective of this project is modelling a vortex conflict detection system that warns the RPAS if it will cross a vortex area which will make it turn upside down, with enough time to change its trajectory in order to do not be in a hazard situation like that.

In order to accomplish this objective, a simple wake vortex model and two collision models will be created. The first collision model will only consider the commercial airliner to fly straight and levelled as most of the time, in upper airspace, airliners fly in that way. Nevertheless, at cruise altitudes they can also perform turns. For this reason, the second collision model will take also into account the turns that the airliner can execute in order to change its heading.

This project starts with the vortex generation model in Chapter 1. In this chapter is described the vortex model that will be used in the collisions models. It is defined the vortex in the horizontal plane, defining the vortex horizontal extension, its trajectory as a consequence of the lateral wind and also the vortex intensity depending on the atmospheric conditions chosen, allowing to know if the vortex is too much severe for the RPAS or not depending on the rolling moment coefficient obtained in the intensity analysis. In this section, it is also described how the vortex evolves in the vertical plane, defining its downward velocity depending also on the vertical wind and when the vortex stabilises, obtaining in that way the affectation vortex area. In Chapter 2 is described the first collision model, explaining in what consist the algorithm created. All the steps in the vertical, horizontal and in the vortex intensity

analysis are explained. Furthermore, in this chapter is explained what type of simulation is performed in order to prove that the algorithm works for all the possible cases from where the RPAS can encounter the vortex. At the end of this section there are some results that affirms the correct functionality of the collision model. In Chapter 3 is explained the second collision model which take into account the turns performed by the commercial aircraft. The algorithm performed is explained such as in Chapter 2, but this time explaining what modifications have been introduced to the first model in order to know where and when the RPAS will encounter the wake vortex after the airliner has performed the turn. At the end of this section, some results of the time the RPAS will need to encounter the vortex, with what headings it will certainly face it and if it will produce to the RPAS a hazard situation or not are explained. Finally, at the Conclusions Chapter are explained the conclusions extracted from the model created in this project.

Moreover, a part from the main body of the project, some bibliography and appendices are added. In Appendix A it is shown the Matlab code of the first collision model, which only considers straight trajectories. Appendix B shows the Matlab code of the second collision model, which takes into account the turns of the airliner.

State of the art

Nowadays, it can be found many studies about the wake vortex generated by aircrafts. However, a lot of the studies that have been performed are about the minimum separation that has to be between two aircrafts when they are in the take-off or landing phases. For instance, in documents [1], [2] and [3] are explained the wake vortex separation minima in the final approach, intermediate approach, departures, landings, crossing and parallel runways.

Furthermore, in the document RECAT-EU [4] is studied another wake vortex categorisation taking into account not only the aircraft's mass but also the wingspan, in order to have more accurate and efficient spacing, increase capacity, but taking always into account the safety factor.

That is a quite interesting analysis, because in this project is not going to be analysed the effect of the wake vortex between two airliners. It is going to be studied the interaction of a wake vortex generated by an airliner and how it affects to a RPAS. Nevertheless, the analysis that will be performed will not be at landing or take off, but Enroute. However, the RECAT-EU could be an interesting document to take into account because until now, the current wake turbulence separation imposed by ICAO [5] was only considering the mass of the of the aircraft, but a RPAS will always have a much lower mass compared to airliners. However, if now the span is considered, as it is known the wingspan of a RPAS is much larger compared with an aircraft of the same mass, so the vortex could have more impact on the RPAS.

For all these reasons, as in this project is going to be determined the minimum separation that must be kept between an airliner and a RPAS, some data of this document will be used.

There are not so many documents about the study of Enroute wake vortex, but after searching, some interesting studies have also been performed and could be interesting to know, so they can be useful in order to perform this project.

A study in [6] about the wake vortex turbulence has been carried out which explains that it endangers greatly the flight safety, because that aerodynamic swirl, which is produced behind the aircraft in flight, stays in free atmosphere for some time. Furthermore, the heavy and clean aircraft, with retracted wheels and wings mechanism, which is the configuration in which airliners will be in Enroute, at low speeds generate the most intensive wake vortex and flying into it can be an unpleasant and a hazard situation.

What is more, additional sophisticated equipment in new modern aircraft improves navigation, communication and steering of the aircraft, however it does not insure it from flying into an invisible turbulent vortex and that is the reason why in [6] is explained the vortex generation, its nature and factors that influence its intensity and duration.

From this document can be extracted that the main cause of the wake vortex turbulence behind a flying aircraft lies in the formation of aerodynamic trailing vortices which are generated due to the circulation around the airfoils. In

addition, there is also the air mass in the jet current behind the engine, but that is relatively small and at a certain distance is practically inexistence.

Therefore, from this paper it will be useful to take into account the calculation of the vortex circulation, which takes into account the weight of the aircraft that generates the vortex, the air density, the airspeed, and the span between the vortex axes, which is obtained doing the product of the aircraft span and a factor of 0.8.

Furthermore, in this document is also discussed the parameters of the wake vortex turbulence, which is also very important in order to define in this project the behaviour of the vortex. From here, it will be extracted the downwash velocity caused by the vortices, which will be applied until the vortex finally stabilises and floats, because their density comes into equilibrium with that of the surrounding air.

Another essential factor and probably the most important one, in order to determine if the wake vortex we are flying into is strong enough to produce a hazard situation, is the intensity of the vortex. This intensity is totally associated to the induced rolling moment which can exceed the roll control of the encountering aircraft, in this case a RPAS, and the capability of an aircraft to counteract the roll imposed by the vortex primary depends on the wingspan and the control responsiveness of the encountering aircraft.

A remarkable document that talks about the metric of the wake turbulence and makes and important emphasis in the Rolling Moment Coefficient (RMC) [7] is the one that it will be used in order to compute the intensity of the wake vortex in this project.

In that document, a simple wake vortex encounter severity metric is proposed. The metric proposed in this paper is for the case of an encounter, by a follower aircraft, having an elliptical wing chord distribution, with a wake vortex, that is centered on the follower wing and that has a Burnham-Hallock circulation distribution with an effective core parameter equal to 3.5% of the generator wingspan. In addition, the metric was assessed based on experimental flight test data results of a Wake Vortex Encounter (WVE) performed by Airbus.

Once the intensity value is obtained from [7], another significant paper where is defined if this value of intensity produces a hazard situation or not will be applied in order to close the intensity analysis. Therefore, in our model will be used [9] to end with Enroute vortex conflict detection system, knowing if there will be conflict with the vortex or not.

Currently, there are also studies that their aims are analysing the risk of wake vortex encounters also in the upper airspace and not near an airport for the landing and take-off phases, in order to understand the risk of the wake vortex in the Enroute flight phase.

Moreover, these kind of studies are taking more interest because with the introduction of the very light jets, or for instance the RPAS, which are susceptible to wake turbulence, along with the super heavy aircraft, such as the

A380, which are powerful generators of wake vortex, the disparity of aircraft types in terms of weights and size has increased.

For this reason, in document [10] they want to quantify the risk that can be. To do it a simulation framework has been defined that makes use of recorded historical surveillance data and computes the wake vortex trajectory and decay for each aircraft.

The wake vortex trajectory has been discretized into seven elements, each defined by a certain volume. The sizes of these volumes are based predominantly on the uncertainty of the aircraft position. Hence, they define two areas: the wake vortex hazard area and the wake vortex habitation area. The wake vortex hazard area is typically of the order of the size of the aircraft. However, the reason why is created the wake vortex habitation area, is because the uncertainty of aircrafts position. In order to facilitate an assessment of potential infringements, both areas are conservatively approximated to rectangles.

What is more, it is also discussed how the wake vortex descents with respect to the trajectory of the airliner and also affirmed that during the first 30 seconds the strength of the vortex is independent of the atmospheric conditions.

Finally, after analysing all the information that has been found about wake vortex from the documents explained above, in order to create an Enroute vortex conflict detection for RPAS operations, which nowadays has not been analysed yet, it will be used some information from the documents explained before in order to carrying it out.

CHAPTER 1. VORTEX GENERATION MODEL

This section presents the designed en-route vortex generation model based on the contributions presented in [6, 7, 10]. A separated dimension analysis will be performed thus splitting the model in two different submodels; one for the horizontal dimension and the other for the vertical one.

The horizontal dimension model will take into account a subset of the dynamic characteristics of the generating aircraft plus the effects of the horizontal wind velocity to calculate the hazard area in which the wake vortex is active. Conversely, the vertical dimension model will focus on the vortex natural dynamic behaviour as described in [6] but also taking into account the vertical component of wind velocity. This way, the vertical displacement of the hazard area is determined. Moreover, to determine the intensity of the vortex in every location of such area, the vortex intensity model presented in [7] will be used. Doing so, the wake vortex will be completely characterized.

1.1. Horizontal plane

As a starting point, the horizontal model will be defined as in [10]: a straight line behind the trajectory of the airliner. Nevertheless, this line will not be right fixed following the direction of the trajectory of the aircraft but will vary its position depending on the velocity (i.e. intensity and direction) where the wind is blowing. Therefore, the vortex will not follow the same trajectory of the aircraft except in case of no crosswind. In this situation, the wake vortex will be strictly behind the aircraft.

To determine the trajectory the vortex will perform, the proposed model will be constructed on the following assumptions. On one hand, as stated in [6], the most important cause of wake vortex turbulence behind a flying aircraft lies in the formation of aerodynamic trailing vortices, which are consequence of the circulation around the airfoils. Nonetheless, there is also the air mass from the jet current behind the engine, but that is relatively small and at a certain distance it can be neglected. For this reason, this model will only consider the circulation of the aerodynamic airflow of the trailing vortices. Therefore, aircraft vortex wakes are, in fact, two parallel, rapidly rotating, spiral tubes of air up to 35 feet in diameter, trailing downstream, such as in figure 1.1.

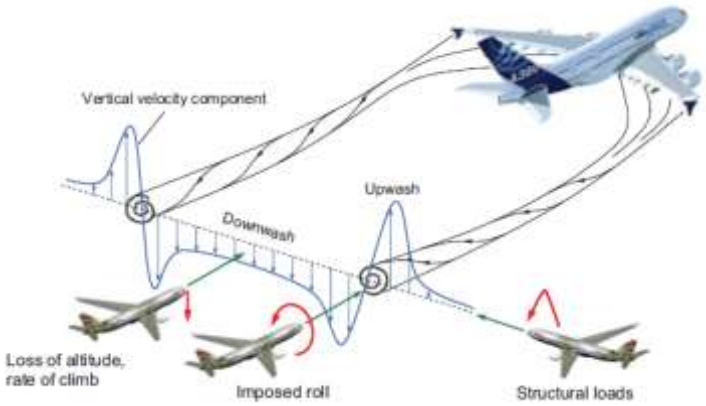


Figure 1.1. Hazard potentials of trailing vortices [11]

Two types of effects can influence the aircraft going through the wake vortex: upwash and downwash effect. In the inner part of the vortex the downwash will take part, which will produce a loss of altitude and rate of climb on a potential follower aircraft. One of the main risks of this situation is the potential collision with an aircraft flying strictly below the follower aircraft. In the outer part of the vortex the upwash effect is found, which will affect the structural load factors. Although, the most dangerous situation will be if these two effects are found together, producing a rolling moment to the encounter aircraft and a possible loss of control.

On the other hand, in [6], authors also state that, at cruise altitudes, the wake vortex can be considered active from 10 up to 40 NM behind the generating aircraft.

As an average, the wake vortex will need around 2-2.5 min to decay and afterwards it will remain at a constant altitude. If it is considered that it remains active a total time of 5 min, taking into account the decay phase, with an aircraft flying at 500 kt will mean that it would have travelled 40 NM and that is what in [6] is affirmed. As a result, the model proposed will consider the vortex to stay active 5 minutes. Nevertheless, this time refers to the total time the vortex needs to disappear due to viscosity, what means that if the intensity is not enough to turn the RPAS upside down, it will not be necessary to go further than 40 NM behind the aircraft to be safe.

About intensity, it will be taken into account the improved Rolling Moment Coefficient (RMC) proposed in [7] to compute a value of intensity. Then it will be able to know if the vortex produces a hazard situation or not.

The vortex generation model can be described, as follows:

Firstly, the initial position of the airliner, speed, heading and altitude at which the airliner is flying are assigned. With all this information, an infinite line can be created, as shown in figure 1.2. It has to be imagined the airliner is being seen from above.

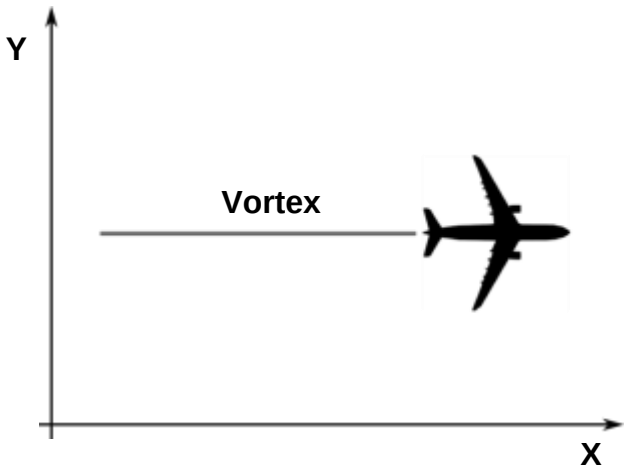


Figure 1.2. Trajectory of the vortex without wind

Using a certain time and considering the velocity constant, applying the rectilinear and uniform motion equation, two points are obtained:

$$x_{1_{airliner}} = x_{0_{airliner}} + v_{x_{airliner}}(t - t_0) \quad \textbf{(1.1)}$$

$$y_{1_{airliner}} = y_{0_{airliner}} + v_{y_{airliner}}(t - t_0) \quad \textbf{(1.2)}$$

Where:

$x_{1_{airliner}}$: is the new *x* position of the airliner [NM]

$y_{1_{airliner}}$: is the new *y* position of the airliner [NM]

$x_{0_{airliner}}$: is the initial *x* position of the airliner [NM]

$y_{0_{airliner}}$: is the initial *y* position of the airliner [NM]

$v_{x_{airliner}}$: Airliner speed in the *x* direction [kt]

$v_{y_{airliner}}$: Airliner speed in the *y* direction [kt]

t : Elapsed time [h]

t_0 : Initial time [h]

With this information the vector of the line (1.3) can be computed:

$$\overrightarrow{u_{airliner}} = \left(x_{1_{airliner}} - x_{0_{airliner}}, y_{1_{airliner}} - y_{0_{airliner}}\right) = (u_{1_{airliner}}, u_{2_{airliner}}) \quad \textbf{(1.3)}$$

Afterwards, with the initial point, the new one computed and the vector, the straight line equation (1.4), which will represent the trajectory of the airliner, is created. In figure 1.2 is shown as the grey line.

$$y = \frac{x - x_{0_{airliner}}}{u_{1_{airliner}}} \cdot u_{2_{airliner}} + y_{0_{airliner}} \quad \textbf{(1.4)}$$

Where:

y: is the new y position of the airliner [NM]

x: is the new x position of the airliner [NM]

$u_{1_{airliner}}$: is the x component of the vector of the straight line [NM]

$u_{2_{airliner}}$: is the y component of the vector of the straight line [NM]

The trajectory obtained will be the one the vortex will follow if no crosswind or no wind is considered. However, in this model the wind is an element to consider, hence its velocity and direction to where is blowing will be defined. As a consequence of taking into account the wind that can be at cruise altitude, the wake vortex generated by the airliner will not follow the trajectory of the airliner anymore. It will be displaced from this path due to crosswind, such as in figure 1.3.

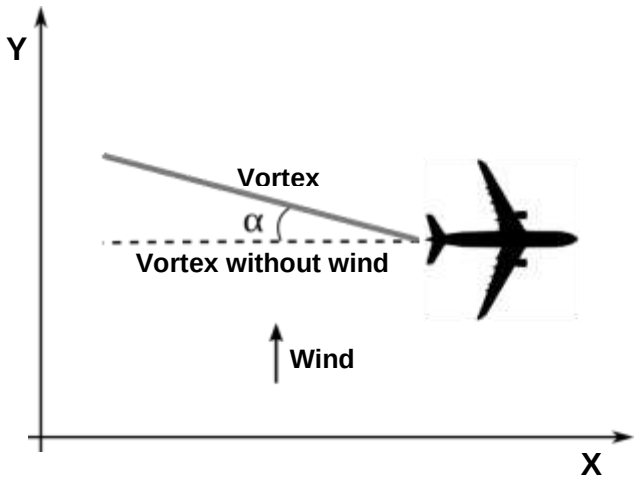


Figure 1.3. Trajectory of the vortex displaced by the wind

At figure 1.3 the wind is blowing from 180º, making the vortex to be displaced from its initial position. If it was blowing from 90° it will be head wind and the trajectory of the vortex would be the same as in figure 1.2. To consider this variation of the vortex trajectory, the wake vortex deviation angle shown in figure 1.3 must be computed with equation (1.5):

$$\alpha = \text{arctg}\left(\frac{\omega_y}{\omega_x + v_{x_{airliner}}}\right) \quad \textbf{(1.5)}$$

Where:

α: is wake vortex deviation angle [rad]

ω_x: is wind speed in the *x* direction [kt]

ω_y: is wind speed in the *y* direction [kt]

With all these calculations, the trajectory the wake vortex will perform depending on where the wind is blowing to is defined.

In terms of wake vortex extension, as described before, from [6] it is explained how long can be extended as an average. Nevertheless, this extension will also depend on the airliner speed, because if it goes at higher velocities it will produce more vortex in less time, therefore the extension of wake vortex will be greater. As an assumption in this model, the wake vortex will remain 5 minutes from the time it has been generated until it vanishes. With this time and the velocity at which the airliner is flying, the extension the wake vortex will occupy is computed with equation (1.6). Hence, the extension of the vortex will not be an average value and will depend on the airliner speed.

$$D_h = V_{airliner} \cdot t \quad \textbf{(1.6)}$$

Where:

D_h: is horizontal vortex extension [NM]

t: is the time the vortex remains [h]

In addition, the vortex area that occupies in terms of wing span must be also analysed.

In [10] two areas are defined: wake vortex habitation area and wake vortex hazard area. The reason why in [10] has been created the wake vortex habitation area, is because the uncertainty of aircrafts position. For that reason, both areas are approximated to rectangles. Nonetheless, in this model the wake vortex habitation area will not be used, because it is considered there is not going to be too much uncertainty. Only the wake vortex hazard area will be used, which is typically of the order of the size of the aircraft as can be appreciated in figure 1.4.

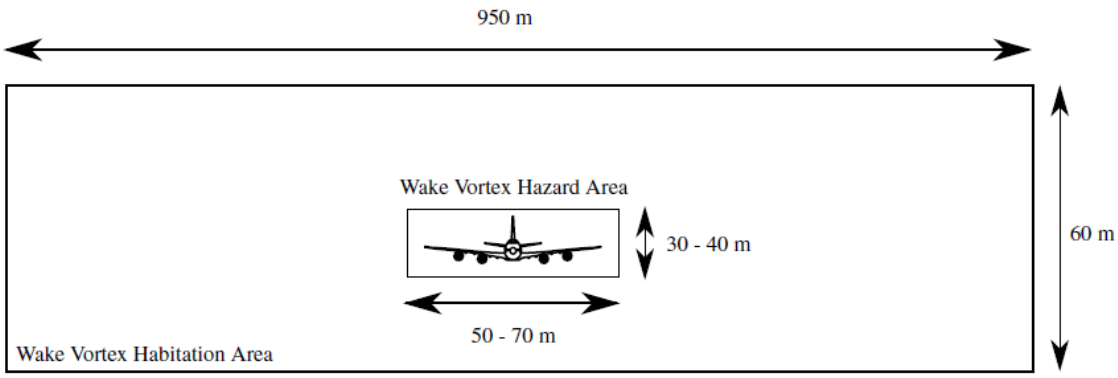


Figure 1.4. Definition of the wake vortex hazard area and wake vortex habitation area [10]

Therefore, the area where the vortex can stay behind the airliner will be a similar one to the dimensions of the airliner. This rectangle will be extended behind the airliner as seen in figure 1.5.

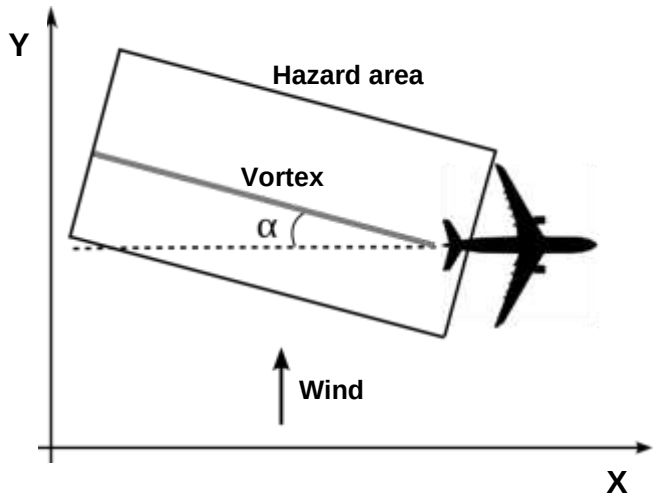


Figure 1.5. Wake vortex hazard area extended behind the airliner

The last aspect to characterise the wake vortex generation in the horizontal plane is its intensity. Intensity is probably the most important factor in the vortex generation model, because it defines if it is too dangerous due to its capacity of turning upside down any aircraft, or if it has lost a lot of intensity and it is not able to produce a hazard situation. In order to define the intensity of the vortex, the Rolling Moment Coefficient based on [7] will be calculated. Equation (1.7) is the one that will be used to compute it.

$$RMC = \frac{\Gamma_{tot}}{V_f b_f} \cdot \frac{AR_f}{AR_f+4} \cdot F\left(\frac{b_l}{b_f}\right) \quad \textbf{(1.7)}$$

With $F\left(\frac{b_l}{b_f}\right) = 1 - 2\left(2r_c\frac{b_l}{b_f}\right)\cdot\left(\sqrt{1+\left(2r_c\frac{b_l}{b_f}\right)^2} - \left(2r_c\frac{b_l}{b_f}\right)\right)$ and $r_c = 0.035$

Where:

Γ_{tot} : is the vortex circulation [m^2/s]

V_f : is the follower flight speed [m/s]

b_f : is the follower span [m]

b_l : is the airliner span [m]

AR_f : is the follower wing aspect ratio [-]

r_c: is the core parameter [-]

Moreover, computing the intensity of the vortex with equation (1.7), some assumptions from [7] are taken into account:

- Centered wake vortex encounter: The vortex center is aligned with the follower wing center. Consequently, the maximum induced rolling moment is obtained.

- Burnham-Hallock vortex circulation distribution with parameter r_c=0.035: It is the most widely used model about circulation distribution for wake vortex applications. It concerns to a circulation distribution model with a core parameter that has a velocity equal to zero at r=0, then increasing to reach a maximum at the effective core radius and then decreasing.

- Elliptical chord distribution assumed for all followers: For the sake of simplicity, it has been considered that the follower wing chord follows an elliptical distribution.

- Effective lift slope correction obtained from solving Prandtl equation for the case of a wake vortex encounter (WVE): The well-known Prandtl correction is not valid for the wake encounter case. What is done is correcting the metric obtained assuming a profile lift slope equal to 2π, using a corrector factor that only depends on the aspect ratio. The effective lift slope should be then computed using the aspect ratio based on the half of the wing. Next, using the Prandlt correction and the half wing aspect ratio is equivalent to use a value of C=4, where C is the correction parameter. Furthermore, this value is confirmed to be robust when changing the wake vortex encounter parameters.

Nevertheless, to be able to compute the Rolling Moment Coefficient, which will determine the intensity of the vortex, it is necessary to know the vortex circulation (Γ_{tot}) value. This parameter is the one that will vary with time and will make the Rolling Moment Coefficient change depending on the time the vortex was generated.

There are different wake prediction models that can be used to define how the vortex circulation decreases with time. From [12] is extracted three types of models:

- Sarpkaya

- Deterministic 2-phase (D2P)

- TASS Driven Algorithm for Wake Prediction (TDAWP)

Both D2P and TDAWP account for two-phased vortex decay, which correspond to semi-empirical wake prediction models that have been formulated from guidance provided by large eddy simulations (LES). The Sarpkaya is also a semi-empirical wake model, but this one predicts wake vortex decay as a function of atmospheric turbulence and stratification. Moreover, unlike the models developed from LES, they predict a rate of decay that is initially large and diminishes with time.

The Sarpkaya model assumes two parallel vortices that decay and descend at equivalent rates. The model has one prognostic equation governing circulation and contains terms for decay due to ambient stratification and turbulence. Vortex descent is determined by using the predicted circulation and a diagnostic relationship for vortex separation.

D2P is the deterministic portion of the Probabilistic 2-Phase (P2P) model which is described in [13],[14],[15]. As in Sarpkaya's model, two parallel vortices are assumed to decay and descend at equivalent rates. However, in D2P circulation decay is governed by an algebraic relationship representing the decay of a simple potential vortex. This algebraic relationship incorporates effects from stratification and ambient turbulence. Similarly, the vortex descent rate is empirically diagnosed from the circulation. The predicted 5-15 m average circulation in D2P is characterized by two phases of decay: the first being a diffusion phase followed by a more rapid rate of decay as indicated in LES experiments.

The TDAWP model has separated prognostic equations for vortex descent rate 5-15 m average circulation. The formulation is driven by parametric studies from LES using TASS [16],[17]. Furthermore, the TDAWP formulation also includes the effects of crosswind shear on vortex descent rate, thus allowing the prediction of vortex tilt and the change in lateral separation due to crosswind. Nevertheless, currently the effects of crosswind shear on circulation decay are not treated in TDAWP nor Sarpkaya nor D2P.

Moreover, from [8] are extracted three graphics that show a comparison of how the circulation varies with time for fast-time out-of-ground-effect wake prediction models, which are Sarpkaya, D2P and TDAWP (figure 1.6).

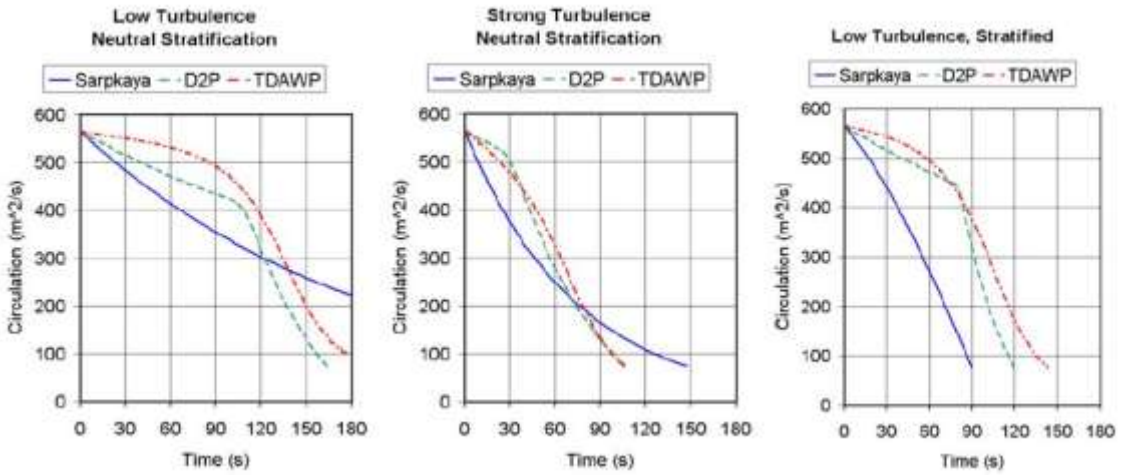


Figure 1.6. Circulation vs time comparison for fast-time out-of-ground-effect wake prediction models

From figure 1.6 can be observed the prediction for circulation depending on the model used. There are large differences between the Sarpkaya model and the D2P and TDAWP models. The circulation decay predicted from the Sarpkaya in the neutral stratification has a convex shape with an initially rapid decay

followed by a slower rate of decay at later times. Whereas, the other two models start with a slow rate of decay but afterwards is followed by a more rapid descent in the circulation value.

Although, linear models, such as the Sarpkaya one, have an advantage of simplicity, theoretical justification for a linear circulation decay is weak when applied to aircraft wake vortices. Therefore, the Sarpkaya model is discarded to be used in our model of circulation. Between the TDAWP and D2P models, it has been chosen the D2P to compute in our study the value of circulation in function of the time the vortex was generated. It has been finally chosen the D2P model because what was being search was a model which gives a value of circulation similar to reality (figure 1.7) and from [12] is affirmed that D2P is the one to obtain better scores, compared with the other models, especially for lateral position and altitude.

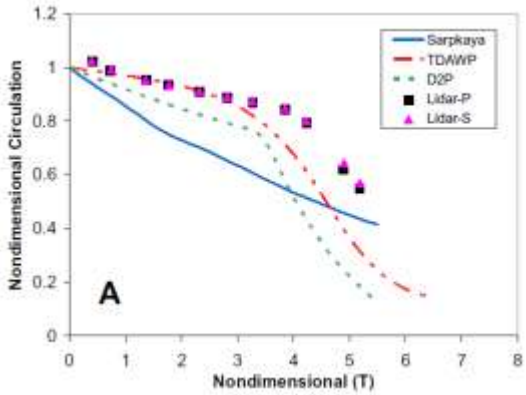


Figure 1.7. Wake measurements obtained by Pulsed Lidar

Ones it has been chosen what model will be used to compute the circulation value, it must be parameterised in function of the time. To do it, the graphics of figure 1.6 will be used. Each graphic represent a different atmospheric situation, taking into account turbulence and stratification. For this reason, the three situations will be considered in our model. Therefore, three different equations must be obtained for each condition. To obtain them, each graphic has been parameterised, several points are extracted from each one and using MATLAB it can be obtained the equation of each atmospheric situation representing circulation in function of time. Figure 1.8 corresponds to low turbulence and neutral stratification, figure 1.9 is related with strong turbulence and neutral stratification and figure 1.10 is about low turbulence and stratified, which corresponds to a stable atmosphere. The black dots correspond to the extracted data from graphics of figure 1.6 and the blue line of the three figures correspond to the regression line that is obtained from the black points, getting equations (1.8), (1.9) and (1.10) of circulation for the different atmospheric conditions.

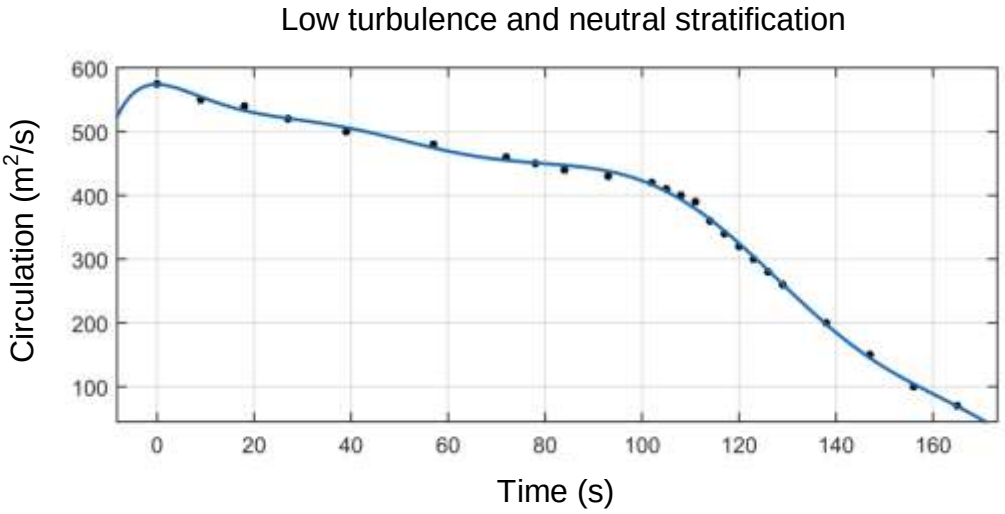


Figure 1.8. Wake vortex circulation vs time with low turbulence and neutral stratification

$$\Gamma_{tot}(t) = P_1 t^9 + P_2 t^8 + P_3 t^7 + P_4 t^6 + P_5 t^5 + P_6 t^4 + P_7 t^3 + P_8 t^2 + P_9 t + P_{10} \quad \textbf{(1.8)}$$

Coefficients:

$P_1 = 3.617 \cdot 10^{-15}$

$P_2 = -2.912 \cdot 10^{-15}$

$P_3 = 9.645 \cdot 10^{-10}$

$P_4 = -1.694 \cdot 10^{-7}$

$P_5 = 1.696 \cdot 10^{-5}$

$P_6 = -0.0009722$

$P_7 = 0.0302$

$P_8 = -0.4331$

$P_9 = -0.09025$

$P_{10} = 574$

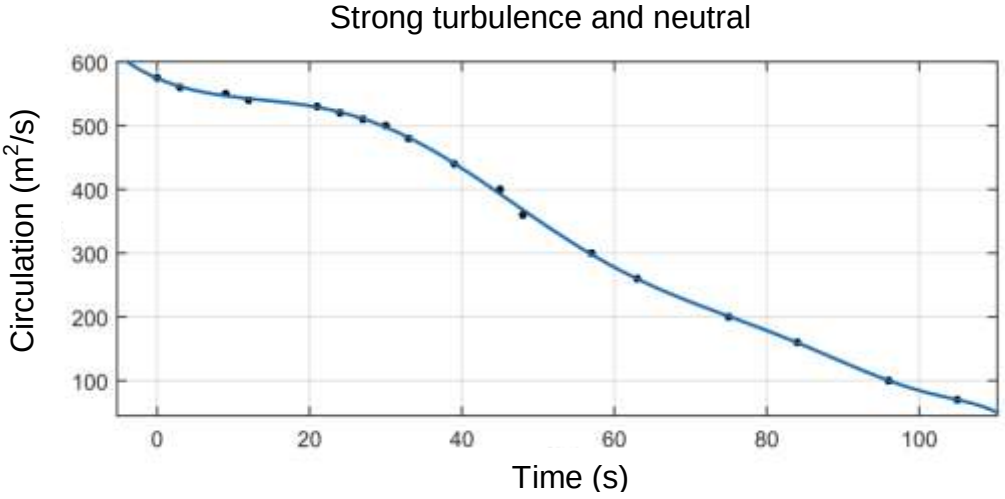


Figure 1.9. Wake vortex circulation vs time with strong turbulence and neutral stratification

$$\Gamma_{tot}(t) = P_1 t^9 + P_2 t^8 + P_3 t^7 + P_4 t^6 + P_5 t^5 + P_6 t^4 + P_7 t^3 + P_8 t^2 + P_9 t + P_{10} \quad \textbf{(1.9)}$$

Coefficients:

$P_1 = -5.357 \cdot 10^{-14}$ $P_6 = 0.0002002$

$P_2 = 2.18 \cdot 10^{-11}$ $P_7 = -0.0102$

$P_3 = -3.399 \cdot 10^{-9}$ $P_8 = 0.\ 3123$

$P_4 = 2.492 \cdot 10^{-7}$ $P_9 = -5.159$

$P_5 = -8.659 \cdot 10^{-6}$ $P_{10} = 574.5$

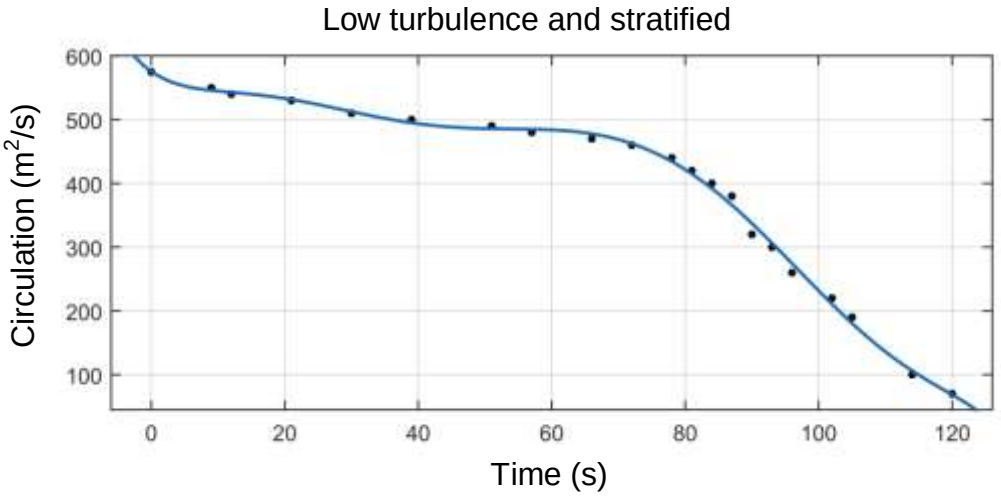


Figure 1.10. Wake vortex circulation vs time with low turbulence and stratified

$$\Gamma_{tot}(t) = P_1 t^7 + P_2 t^6 + P_3 t^5 + P_4 t^4 + P_5 t^3 + P_6 t^2 + P_7 t + P_8 \quad \textbf{(1.10)}$$

Coefficients:

$P_1 = -1.569 \cdot 10^{-10}$ $P_5 = -0.0367$

$P_2 = 6.823 \cdot 10^{-8}$ $P_6 = 0.7101$

$P_3 = -1.138 \cdot 10^{-5}$ $P_7 = -7.335$

$P_4 = 0.0009121$ $P_8 = 575.7$

Consequently, depending on what atmospheric conditions there are, one of the three cases will be chosen. Hence, the circulation value when the RPAS crosses the vortex is known and this value can be introduced at equation (1.7),

obtaining the Rolling Moment Coefficient that will produce the wake vortex, being its intensity defined.

Knowing the vortex intensity value, the last step to perform in order to know if the RPAS can fly trought the vortex is knowing if this RMC can produce a dangerous situation or not. To do that, the RMC of control is computed like in [9]. The RMC of control is the maximum coefficient of roll that can be created by the control system of the RPAS in order to balance the vortex rolling moment and making the RPAS to remain stable. Finally, the RMC generated by the vortex is divided by the RMC of control of the RPAS. If this value is larger than one, it will mean that it is a dangerous situation because the vortex would be able to turn the RPAS upside down and if it is smaller than one, the RPAS could fly without changes in its trajectory because it would be safe for it to fly trought the vortex region.

1.2. Vertical plane

To define the trajectory the vortex will perform in the vertical plane, the proposed model will be based on the following assumptions. In [7] is stated that the wake vortex descends with respect to the trajectory of the aircraft and that it decays with an average speed of 400-500 ft/min. At cruise altitude, vortices usually level off at about 1000 ft below the altitude of the aircraft as their density comes into equilibrium with that of the surrounding air. Decay processes then take over as observed in figure 1.11.

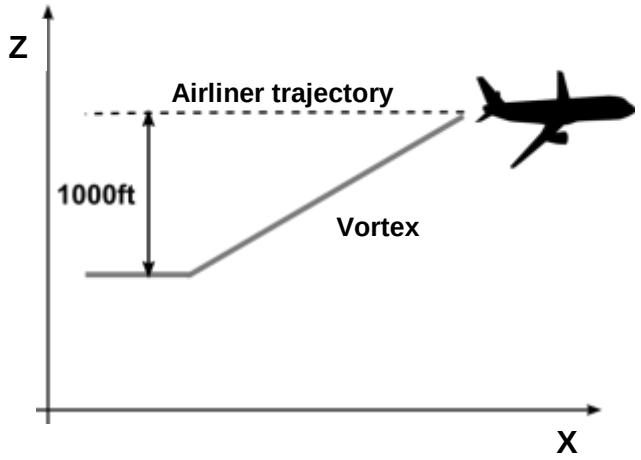


Figure 1.11. Wake vortex stabilises 1000ft below aircraft altitude

Moreover, from [6] is extracted that the rate of circular motion around every vortex acts on the trailing vortices and causes downwash with a velocity determined by equation (1.11).

$$w = \frac{\Gamma_0}{2\pi \cdot L_v} \quad \textbf{(1.11)}$$

Where:

w: is downwash velocity [m/s]

Γ_0: is vortex circulation [m^2/s]

L_v: is span between vortex axes [m]

The vortex circulation (Γ_0) is computed by equation (1.12).

$$\Gamma_0 = \frac{W}{\rho \cdot V \cdot L_v} \quad \textbf{(1.12)}$$

Where:

W: is weight of the generator [$kg \cdot m/s^2$]

ρ : is air density [kg/m^3]

V: is airspeed [m/s]

The span between the vortex axes is distance b' from figure 1.12.

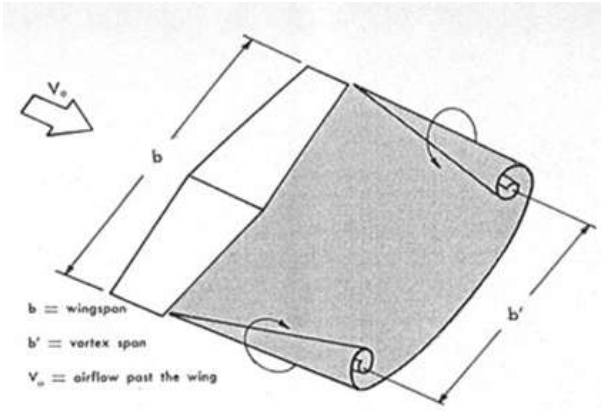


Figure 1.12. Vortex sheet

The span between the vortex axes is determined by the layout of airflow along the wing span, which depends on the shape of the wings and angle of attack. In free air the factor that relates the wingspan and the span between the vortex axes is of 0.8 [6]. Therefore, the span between vortices will be computed with formula (1.13).

$$L_v = 0.8 \cdot b \quad \textbf{(1.13)}$$

Where:

b: is airliner span [m]

Hence, in this model will be used the downwash velocity computed by equation (1.11), instead of using an average value. Consequently, this model will be

taking into account the weight, the velocity and the characteristics of the wing of the vortex generator. That is a very important fact because depending on the type of aircraft which generates the vortex, it will go downwards with a different velocity and it is important to know the time it needs to stabilise in order to make the vertical analysis.

The last assumption that must be taken into account to perform the vertical analysis is not to consider the vortex as a point at a certain altitude, but as volume. Therefore some margins must be applied. As explained at the horizontal plane section, in [10] two areas are defined, but only the wake vortex hazard area will be taken into account. Hence, if we look at figure 1.4 only the inner rectangle will be considered. Therefore, from [10] is extracted that the altitude of the rectangle in the vertical plane will be also similar to the aircraft size.

The intensity of the vortex does not depend on altitude, but it does depend on the time it has been generated. Therefore, it is enough to model the intensity once at the horizontal plane.

Finally, the wake vortex generation model in the vertical part is defined and the behaviour of the vortex will be the one observed in figure 1.13. It will descend with a *w* downwash velocity until it stabilises at 1000ft and it will occupy a certain area in the vertical plane.

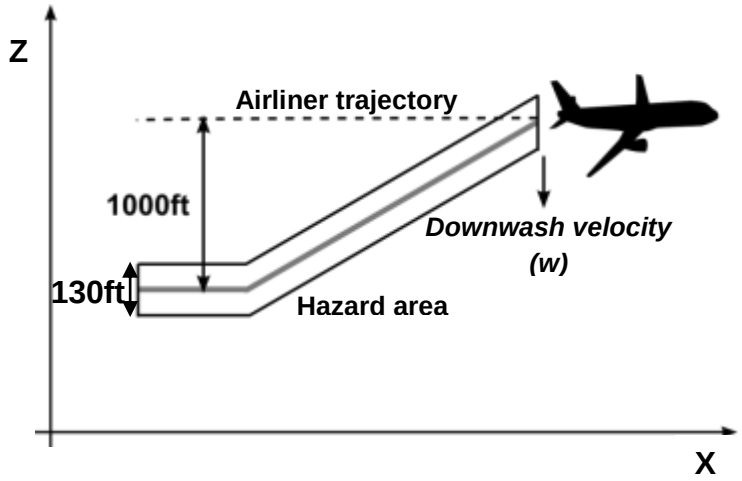


Figure 1.13. Wake vortex generation of the vertical plane

CHAPTER 2. COLLISION MODEL

Once the vortex generation model has been defined, in this section will be described a collision model between the wake vortex turbulence generated by the airliner and a RPAS.

First, the scenario of the model will be exposed, describing the trajectories of each aircraft and all the assumptions needed to perform the algorithm. After that, all the steps of the collision algorithm will be described in order to detect if a hazard situation could happen or not. In the algorithm, once the vortex is defined, the next step is obtaining the encounter point. After that a vertical analysis will be performed to check if the RPAS is inside the vertical hazard area. If it is inside, a horizontal analysis will be carried out to analyse if it is inside the horizontal vortex affectation area. Finally, the intensity of the vortex will be studied to affirm if there is any risk if the RPAS is inside the global hazard area.

2.1. Assumptions

Some assumptions will be defined in order to carry out the collision model in this project. The RPAS and the airliner will follow a rectilinear and uniform motion. Both will fly at constant speeds and altitudes. Wind will be considered in the model and its speed will also remain constant.

The airliner considered in this collision model will be an A320 [18], whereas the RPAS could be the Global Hawk (RQ-4) [19] or the Ikhana (MQ-9) [20]. Another parameter that can be chosen is the atmospheric condition at which it is wanted to analyse the possible vortex conflict.

With the purpose of assuring that the model works correctly, a first fiction scenario will be created. The airliner could fly to any direction, but it has been decided that it will fly with a heading of 180° and the RPAS will come from all the directions, such as in figure 2.1. Therefore, it will be able to study all the conflict possibilities and see that the model will be useful all the cases.

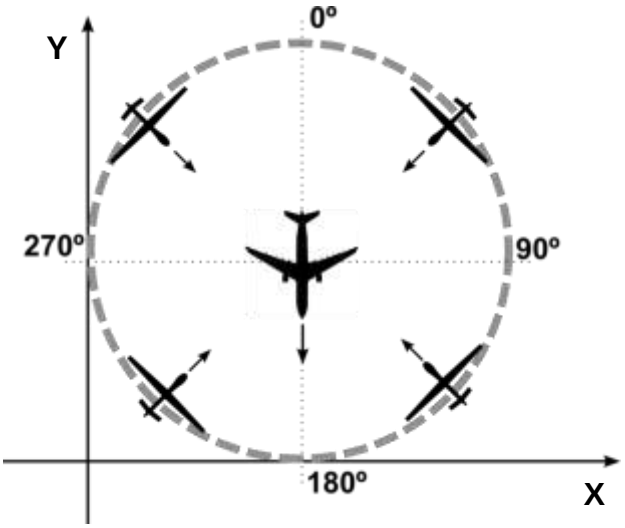


Figure 2.1. Scenario to check the model

In addition, another assumption in the conflict model is that once the intersection point between trajectories is found, that point will be the [0,0], the time will be initialised to zero and at this time the RPAS will be allocated in the intersection between the airliner and RPAS trajectories.

2.2. Description of the algorithm

In the collision algorithm, the first step is assigning the initial position, speed, heading and altitude at which the airliner is flying. With all this information, as explained in section 1, using a certain time and considering the velocity constant, if the rectilinear and uniform motion equation is applied, two points are obtained using equations (1.1) and (1.2).

With these two points, a vector of the line (1.3) is created and with (1.1), (1.2) and (1.3), equation (1.4) that refers to the trajectory that will perform the airliner is defined.

Secondly, the same procedure must be done with the RPAS, obtaining its initial position, speed, heading and altitude at which is flying. With all that and a certain time, two points are computed using rectilinear and uniform motion equations (2.1) and (2.2):

$$x_{1_{RPAS}} = x_{0_{RPAS}} + v_{x_{RPAS}}\,(t - t_0) \qquad \textbf{(2.1)}$$

$$y_{1_{RPAS}} = y_{0_{RPAS}} + v_{y_{RPAS}}(t - t_0) \qquad \textbf{(2.2)}$$

Where:

$x_{1_{RPAS}}$: is the new *x* position of the RPAS [NM]

$y_{1_{RPAS}}$: is the new *y* position of the RPAS [NM]

$x_{0_{RPAS}}$: is the initial *x* position of the RPAS [NM]

$y_{0_{RPAS}}$: is the initial *y* position of the RPAS [NM]

$v_{x_{RPAS}}$: RPAS speed in the *x* direction [kt]

$v_{y_{RPAS}}$: RPAS speed in the *y* direction [kt]

t : Elapsed time [h]

t_0 : Initial time [h]

With this information the vector of the line (2.3) can be computed:

$$\overrightarrow{u_{RPAS}} = \left(x_{1_{RPAS}} - x_{0_{RPAS}},\, y_{1_{RPAS}} - y_{0_{RPAS}}\right) = \left(u_{1_{RPAS}}, u_{2_{RPAS}}\right) \qquad \textbf{(2.3)}$$

Then, with the initial point, the new one computed and the vector, we can create the straight line equation (2.4) which will represent the trajectory of the RPAS.

$$y = \frac{x - x_{0_{RPAS}}}{u_{1_{RPAS}}} \cdot u_{2_{RPAS}} + y_{0_{RPAS}} \quad \textbf{(2.4)}$$

Where:

y: is the new y position of the RPAS [NM]

x: is the new x position of the RPAS [NM]

$u_{1_{RPAS}}$: is the x component of the vector of the straight line [NM]

$u_{2_{RPAS}}$: is the y component of the vector of the straight line [NM]

Once equations (1.4) and (2.4) are obtained, the intersection point between these two infinite straight lines can be computed. If we equate both equations, the *x* value of the intersection point will be obtained. Substituting this value in one of the equations, the value of *y* at the intersection will be acquired. This intersection point will be the point where the RPAS would have to face the wake vortex if there was no wind or if there was not crosswind, as can be observed in figure 2.2. Nonetheless, if there is crosswind, the conflict point would not be this intersection between trajectories. Hence, some more calculation must be done.

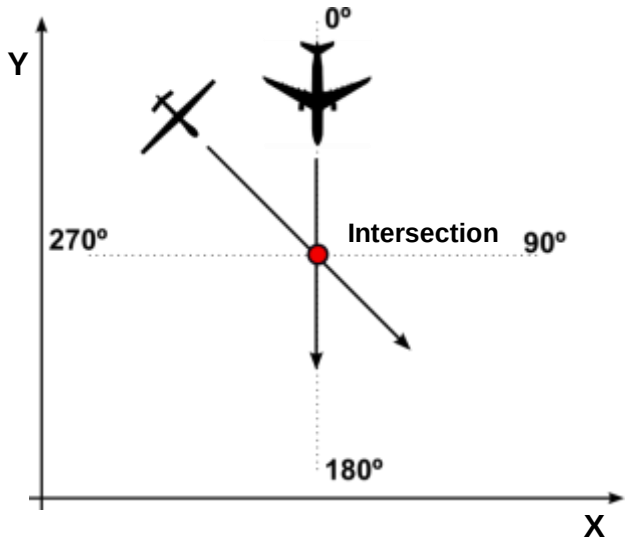


Figure 2.2. Intersection point between RPAS and airliner

At this point, some information that will be used afterwards will be computed. As the initial position of the RPAS and the intersection point are known, the following step is computing the distance between these two points. As the RPAS speed is known and constant, the time that it spends going from its initial position to the intersection between the RPAS and airliner trajectories can be computed ($t_{int,\ RPAS}$). For the airliner it will also be calculated the distance that it has flown up to the intersection point and with this data and its constant

velocity, the time it has needed to reach the intersection point can be computed, too ($t_{int,\ airliner}$). In figure 2.3. can be observed these times.

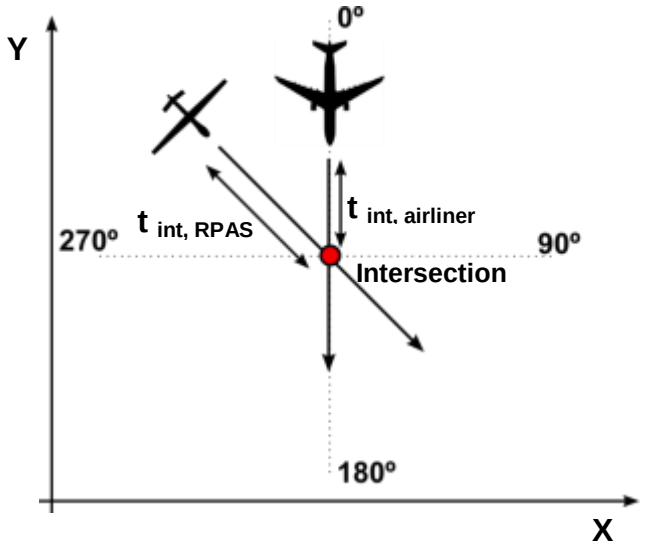


Figure 2.3. Times required to reach the intersection point

To know the airliner position when the RPAS is at the intersection point, the time required by the RPAS to reach this point will be used together with the airliner velocity.

Next, it is studied if the RPAS is ahead or behind the airliner at the intersection of both trajectories. If the RPAS is ahead the aircraft when it has reached the intersection point, it will not have any conflict with the wake vortex turbulence generated. For that reason, that situation will be safe as shown in figure 2.4.

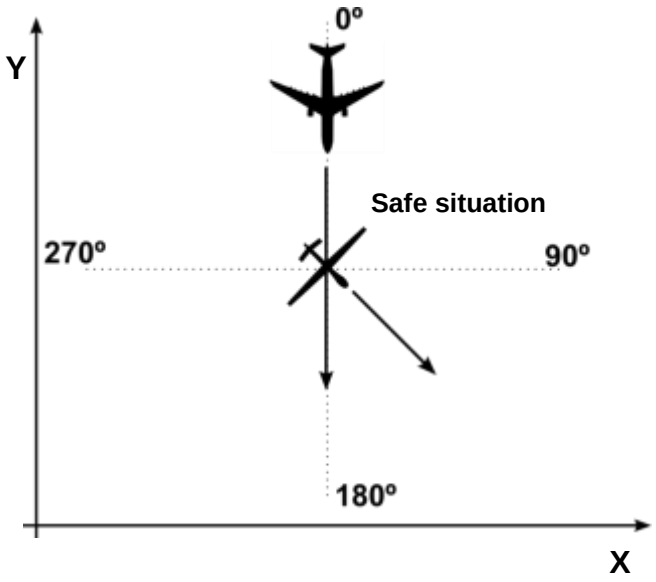


Figure 2.4. Safe situation, RPAS ahead the airliner

Nevertheless, if the RPAS is behind the airliner, such as in figure 2.5, the analysis is not finished because the RPAS could be inside the affectation area of the vortex (grey line in figure 2.5). Hence, more analysis must be done in order to ensure safety.

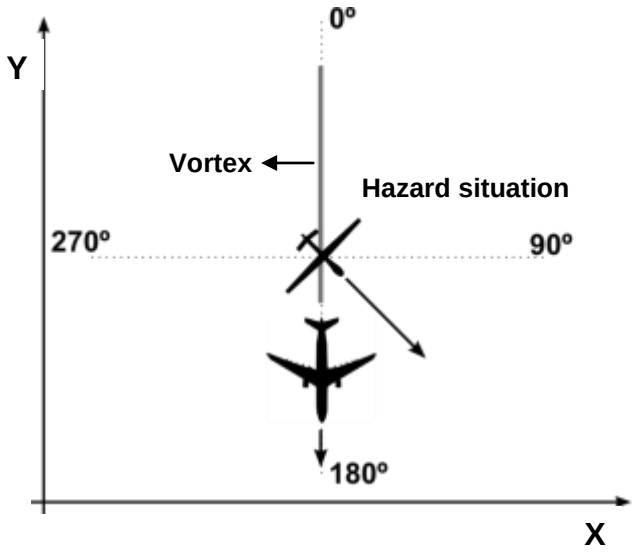


Figure 2.5. Hazard situation when the RPAS is behind the airliner

Once we have arrive at this point of the algorithm, as there is the possibility to come across the vortex in some moment of the RPAS trajectory, the course of the wake vortex turbulence must be defined. As in this model the wind effect is taken into account, it has to be computed the path that will follow the wake vortex in function of the wind direction and intensity. Moreover, the extra time required to encounter the vortex from the intersection point until the RPAS encounters it, must be also calculated.

In order to analyse if the possible conflict is really a hazard or not, the time and the position of the RPAS and the airliner are initialised to zero. When time is zero, the RPAS is at the intersection point which has been considered to be at position [0,0]. Then, it must be computed according to these new initial conditions, where the airliner will be at time equal to zero. As before, it has been computed where the airliner was when the RPAS was at the intersection point, the distance between that position and the intersection is known. Therefore the time (Δt) needed by the airliner to go from the intersection point to its position when the RPAS is at the intersection point is computed.

In addition, the airliner is forced to fly with a heading of 90º, so the headings of the aircraft and the RPAS are rotated in order to make the airliner fly with the heading mentioned. Therefore in figure 2.6. can be observed how the airliner heading changes from 180º to 90º and the RPAS heading changes from 135º to 45º. In both cases to the original headings have been subtracted 90º.

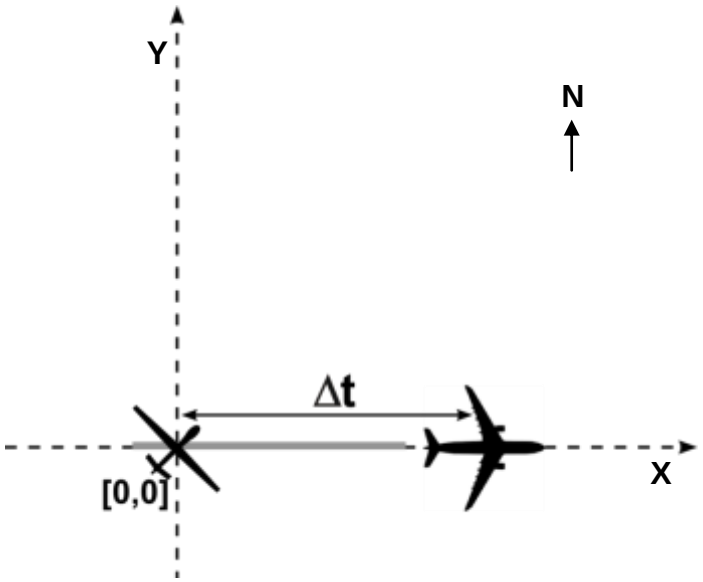


Figure 2.6. Scenario initialised

Afterwards, it is computed the wake vortex deviation angle (α), the angle of the conflict (β) and the encounter angle (γ).

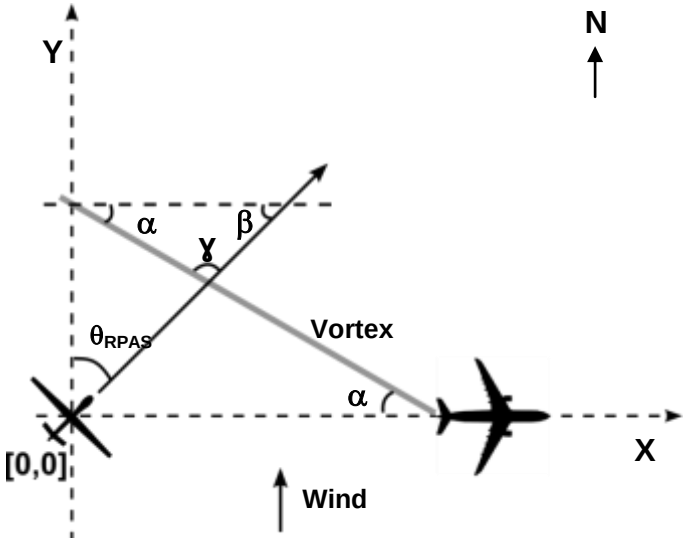


Figure 2.7. Definition of α, β and γ

Using the geometry of the triangle (figure 2.7), α and β can be calculated applying formulas (1.5), (2.5) and (2.6).

$$\beta = 90 - \theta_{\mathrm{RPAS}} \quad \textbf{(2.5)}$$

$$\gamma = 180 - \beta - \alpha \quad \textbf{(2.6)}$$

Where:

β: is the angle of the conflict [º]

θ_{RPAS}: is the RPAS heading [º]

γ: is the encounter angle [º]

α: is the wake vortex deviation angle [º]

Next step will be finding the encounter vortex position. It has to be taken into account that the vortex position that has been obtained at first, is when the RPAS is at the intersection point, but if there is crosswind the wake vortex turbulence will not be in the airliner trajectory. It will have been displaced. Therefore, two situations can be described depending from where the wind is blowing. The RPAS will find the airliner in a certain time after going through the intersection point, as seen in figure 2.8, or the RPAS could encounter the vortex before reaching the intersection between trajectories as in figure 2.9.

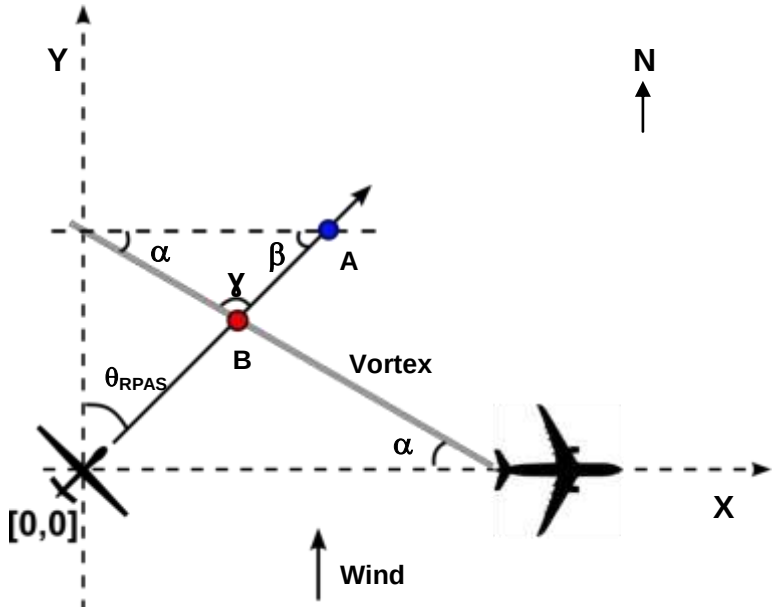


Figure 2.8. RPAS will find the vortex after the intersection point

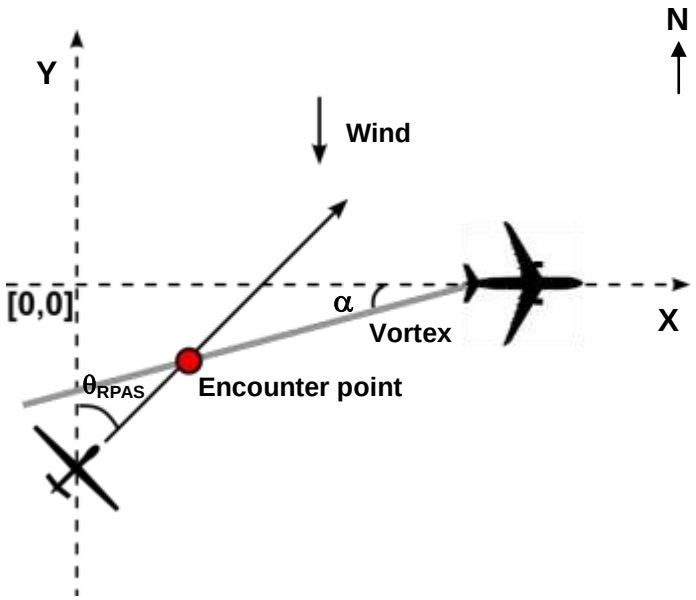


Figure 2.9. RPAS will find the vortex before the intersection point

The first encounter point A from figure 2.8 is based on finding when the RPAS catches the vortex being displaced by the wind in the *y* axis. To compute it, from figure 2.7, the angles α and β are used. The *y* encounter position and the *x* encounter position will be calculated with equations 2.7 and 2.8, respectively.

$$y_{encounter} = x_{airliner} \cdot \tan(\alpha) \quad \textbf{(2.7)}$$

$$x_{encounter} = \frac{y_{encounter}}{\tan(\beta)} \quad \textbf{(2.8)}$$

The time the RPAS needs to arrive to the encounter point can be computed by equation 2.9.

$$t_{enc} = \frac{\sqrt{(x_{encounter} - x_{RPAS})^2 + (y_{encounter} - y_{RPAS})^2}}{v_{RPAS}} \quad \textbf{(2.9)}$$

From figure 2.8 can be observed that the first encounter point A is not possible to be the real one, because it is clearly seen that the RPAS will find the vortex before that calculated point. Hence, point B will be the most possible point to be the encounter one from figure 2.8. The time needed by the RPAS to reach point A is $t_{enc,1}$, but as point A is not the real encounter one, that means the encounter point will be reached a certain time before the calculated one. It will be a smaller time. Therefore, it must be computed the time the RPAS will need to reach the vortex from the intersection point between trajectories.

However, if we consider point B to be the point where the RPAS will get the vortex, which means to move the RPAS a certain time before ($t_{enc,2}$), that would mean that the airliner will have to be move also to that certain time in the past ($t_{enc,2}$), as can be observed in figure 2.10. Then, it will be observed that we will be at the same situation than before. The RPAS would encounter the vortex

before the calculated point. Hence, point B of figure 2.8 could be called A' at figure 2.10 because it will not be the encounter point.

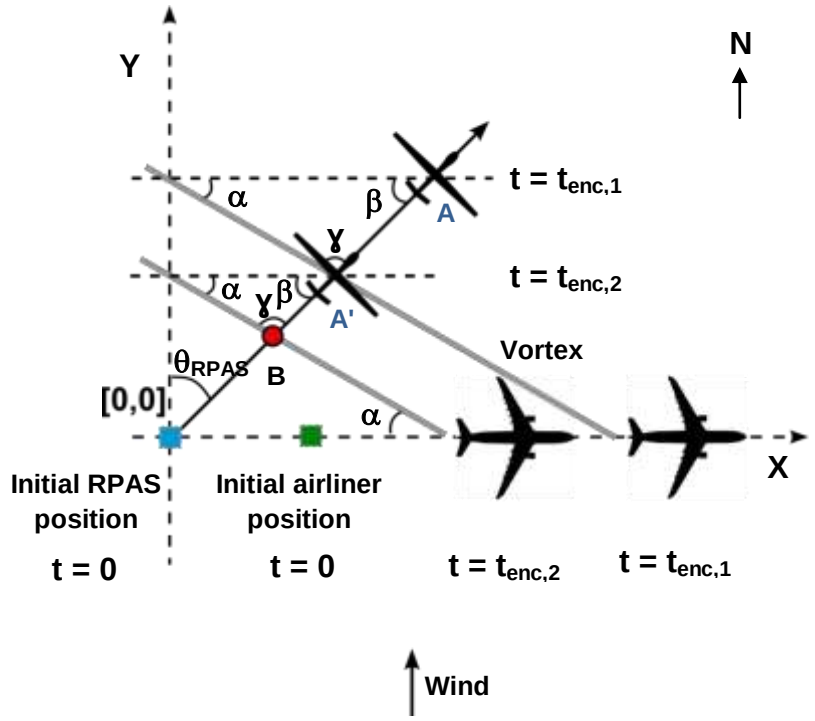


Figure 2.10. Analysis of the time the RPAS need to reach the vortex

As a result, a loop has been created in order to stop the searching of the time that will need the RPAS to get in conflict with the vortex. When this time is similar enough to the value computed before, which will mean that the value converges, the search is stopped and the time is saved.

Knowing the time the RPAS will need to encounter the vortex (t_{enc}), we can collocate the RPAS to the encounter point (figure 2.11) and then it will be able to know at what distance is the RPAS from the airliner. What is more, with this distance and knowing the speed at which the airliner is flying, it can be computed how long ago the vortex has been created (t_{vortex}) and if it is still active or not looking at its intensity.

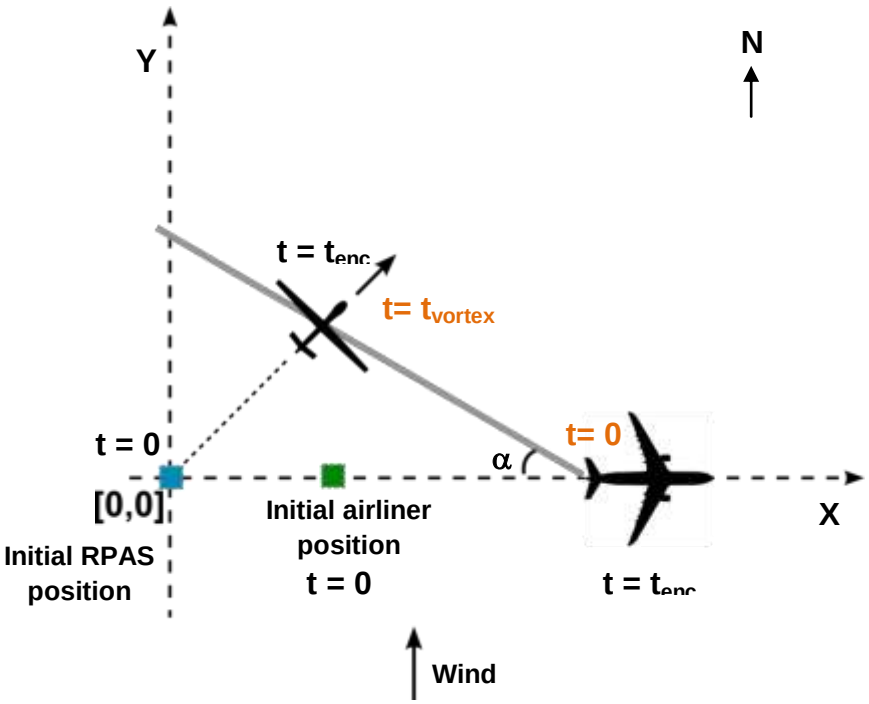


Figure 2.11. RPAS at encounter point

Once it has been defined the position where the RPAS intersects the trajectory of the wake vortex, it has to be analysed if the RPAS will be in a hazard situation or not. To do it, a vertical and a horizontal analysis will be performed. The vertical analysis will be the first one to be carried out because in terms of computations is simpler than the horizontal one. Therefore, if the RPAS is not inside the vertical hazard area of the vortex, it will not be necessary to analyse the horizontal one because it will have been already discarded from the dangerous situation. Hence, no more computation will have to be executed.

However, if the RPAS is inside the vertical hazard area, the horizontal analysis will have to be carried out in order to see if it is inside the horizontal hazard area of the wake vortex or not.

2.2.1. Vertical analysis

The first step is analysing if the RPAS could be inside the vertical area of the vortex affectation and see if it could be in a dangerous situation or if its altitude makes it to be out of the hazard location.

From the previous step the encounter point is known. Consequently, it has been computed the distance that there is between that point and the position of the aircraft. Knowing this distance and the speed at which the aircraft flies, it has been calculated how long ago the vortex was generated.

Afterwards, what must be computed is the time the vortex needs to stabilise. As it is known that the vortex decay stops 1000 ft under the aircraft, dividing this value by the downwash velocity, the time needed to stabilise is obtained. If vertical wind is considered negligible, then the vertical downwash velocity will be the one computed with equation (1.11). But, if vertical wind velocity is not considered zero, then to the downwash velocity computed will be added the vertical speed of the wind if it goes downwards or subtracted if the wind velocity goes upwards, as can be seen in figure 2.12 and 2.13, respectively.

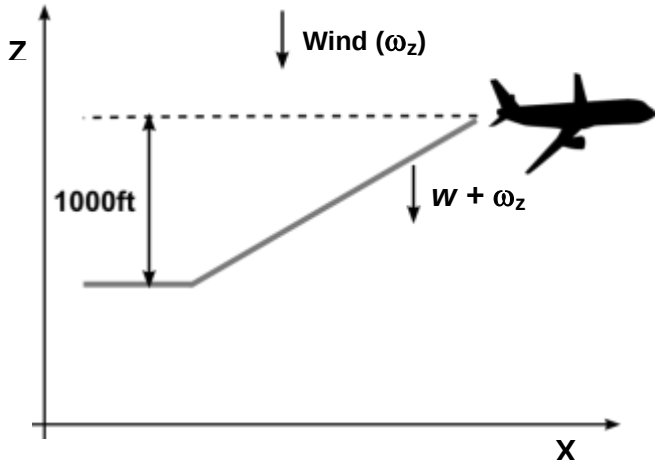


Figure 2.12. Vortex decays at a speed of W+ω_z

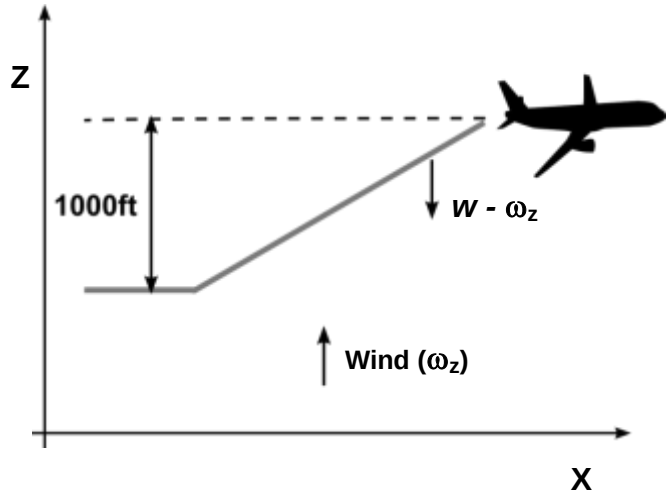


Figure 2.13. Vortex decays at a speed of W-ω_z

The equation used to compute the time the vortex will need to stabilise is (2.10)

$$t_{stable} = \frac{1000}{w \pm \omega_z} \quad \textbf{(2.10)}$$

Where:

t_{stable}: is time vortex needs to stabilise [min]

w: is the vortex downwash velocity [ft/min]

ω_z: is wind speed in the *z* direction [ft/min]

At this point, two situations can be produced:

- When the RPAS arrives at the encounter point and the time that has passed since the airliner has gone through that point is larger than the time the vortex needs to stabilise. In this case, the altitude of the vortex will be 1000ft below the aircraft's altitude.

- When the RPAS arrives at the encounter point and the time that has passed since the aircraft has gone through that point is smaller than the time the vortex needs to stabilise. In this situation, it should be computed the altitude at which the vortex will be.

Once it is obtained the altitude of the vortex, the margins which will form the wake vortex hazard area, must be applied. For an average aircraft, the vertical margin will be of a maximum of 40 meters (130 ft) as seen in figure 1.4. Therefore, to the obtained altitude of the vortex, it should be added 65ft to get the upper limit and subtracted 65 ft to obtain the lower limit of the vortex.

Hence, it must be studied if the altitude of the RPAS when it is flying through this encounter point, is inside these limits or outside. If it is inside the wake vortex hazard area, it could be in a dangerous situation. Although, it must also be studied the horizontal plane and see if it is inside of it or not. If it was inside both hazard areas, the intensity that would have the vortex at that point will have to be studied.

2.2.2. Horizontal analysis

If after performing the vertical analysis, it results that the RPAS is inside the hazard vertical area of the wake vortex, the horizontal analysis must be performed.

Knowing the time the vortex stays until it totally disappears and the speed at which the aircraft is flying, it can be calculated the distance that the wake vortex will be active. Afterwards, it is computed the distance that there is between the encounter point, where is where the RPAS will be, and the airliner.

Finally, these two values are compared. If the distance between the RPAS and the airliner is smaller than the one that the vortex stays, it will mean that the RPAS could be in a hazard situation because it is inside the active area of the vortex (grey line of figures 2.14 and 2.15). Subsequently, the intensity of the vortex is analysed in order to see if is really dangerous or not. Conversely, if the distance between the RPAS and the airliner is larger than the distance the vortex stays active, the RPAS will be in a safe location. In figure 2.14 and 2.15 can be appreciated both situations.

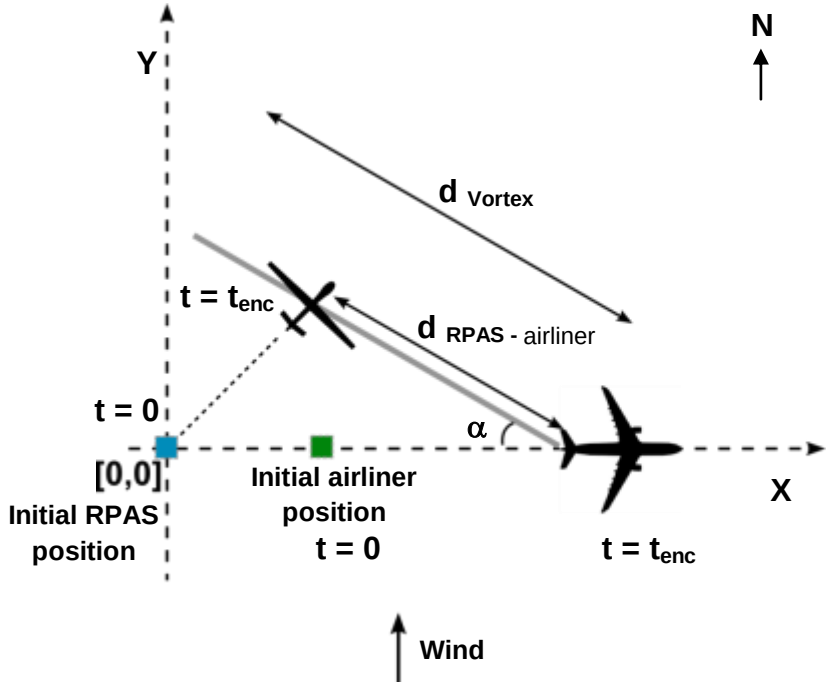


Figure 2.14. Hazard situation: distance RPAS - airliner smaller than the vortex one

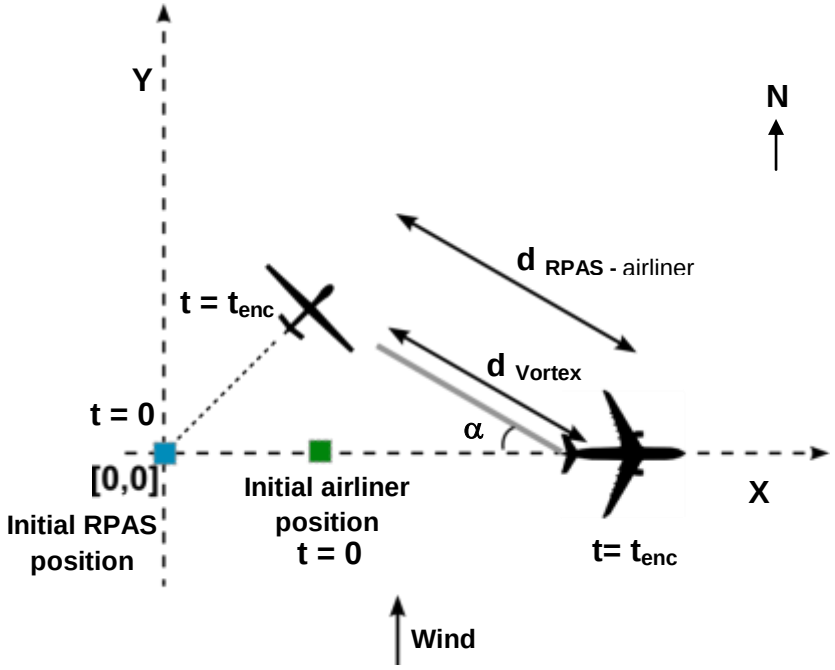


Figure 2.15. Safe situation: distance RPAS- airliner larger than the vortex one

Therefore, if in both analysis the RPAS has been encountered inside the hazard areas of the vortex, the intensity that will have the vortex at the point where the RPAS is must be computed. Afterwards, it will have to be analysed if this value of intensity will make turn the RPAS upside down or if it will not be able to do it and it will only cause some turbulence which will not be dangerous.

2.2.3. Vortex intensity analysis

If the vertical analysis and the horizontal analysis have affirmed that the vortex is inside the hazard area of the vortex, the intensity of it must be studied in order to know if the locality of the RPAS will be safe or if the vortex has too much intensity for the RPAS and will make turn it upside down.

As an input in our collision model, the type of atmospheric conditions must be selected among the three different situation that our model includes: Low turbulence and neutral stratification, strong turbulence and neutral stratification and low turbulence and stratified.

Knowing the atmospheric conditions that are wanted to be simulated, the value of circulation in that atmospheric circumstances can be computed applying the corresponding formula (1.8), (1.9), (1.10) which represents each atmospheric scenario respectively. To be able to compute the circulation, it is necessary to know how long ago the vortex was generated when the RPAS crosses it, which means to compute how long the airliner has passed through the encounter point using equation (2.11)

$$t_{gen} = \frac{d_{vortex}}{V_{airliner}} \cdot 3600 \qquad \textbf{(2.11)}$$

Where:

t_{gen}: is the time how long the vortex is generated [s]

d_{vortex}: is the distance between the airliner and the RPAS [NM]

$V_{airliner}$: is the airliner velocity [kt]

However, the value of circulation obtained by equations (1.8), (1.9), (1.10), depending on what atmospheric condition has been chosen, is based on the graphics of figure 1.6, where the initial circulation was 575 m^2/s. But in our model where the aircraft generator is an A320, the initial circulation will be different. It is computed as explained in section 1.2 using equation (1.12). Therefore, the total circulation obtained will be based on an initial circulation of 575 m^2/s. In order to obtain the value of the circulation in our model, to the value obtained will be subtracted 575 m^2/s and afterwards it will be added the initial circulation computed in our model.

Subsequently, knowing the value of the circulation when the RPAS flies through the vortex, the Rolling Moment Coefficient, which represents the vortex intensity, can be computed using formula (1.7).

Nevertheless, to know if exists a hazard situation when the RPAS is crossing the vortex, it is needed something more than the RMC generated by the vortex. Next step, will be computing the RMC of control. To compute $RMC_{control}$, equation (2.12) will be used.

$$RMC_{control} = -RMC_p \left(\frac{pb}{2U_o}\right) \qquad \textbf{(2.12)}$$

Where,

$RMC_{control}$: is the coefficient of roll created by the control system [-]

RMC_p: is the roll damping coefficient [-]

$\left(\frac{pb}{2U_o}\right)$: is the roll rate [-]

From [9] is defined the roll rate to be 0.07.

The roll damping coefficient is computed using equation (2.13).

$$RMC_p = -\frac{C_{L_\alpha}[1+3\lambda]}{12\ [1+\lambda]} \quad \textbf{(2.13)}$$

Where,

C_{L_α}: is finite wing lift curve slope [-]

λ: is the taper ratio [-]

These both parameters depend on the type of RPAS is being analysed. In our model the type of RPAS used is an input and can be chosen between two types: the Global Hawk (RQ-4) and Ikhana (MQ-9).

The $C_{L\alpha}$ for both types of RPAS has been obtained in the same way, using the graphic of C_L vs α of each RPAS. This type of graphic has been obtained first considering a straight and balanced flight, being able to consider lift equal to weight, obtaining equation (2.14).

$$\frac{1}{2}\rho v^2 S C_L = W \quad \textbf{(2.14)}$$

Where,

ρ: is density [kg/m^3]

v: is velocity of the RPAS [m/s]

S: is the wing surface [m^2]

C_L: is the lift coefficient [-]

W: is the RPAS weight [kg]

Furthermore, the C_L can be expressed as equation (2.15)

$$C_L = C_{L_0} + C_{L_\alpha} \cdot \alpha \quad \textbf{(2.15)}$$

Where,

C_{L_0}: is the C_L value when α is zero [-]

α: is the angle of attack [º]

So, taking two different values from the graphic C_L vs α and using equation (2.15), the value of $C_{L\alpha}$ is obtained for both RPAS. For the Global Hawk its $C_{L\alpha}$ value is 0.105 and for Ikhana is 0.122.

In order to compute the taper ratio (λ) of each RPAS equation (2.16) will be used.

$$\lambda = \frac{c_t}{c_r} \quad \textbf{(2.16)}$$

Where,

c_t: is the dimension of the wing tip [m]

c_r: is the dimension of the wing part in contact with the fuselage [m]

In figure 2.16 can be better understood which are these c_t and c_r dimensions.

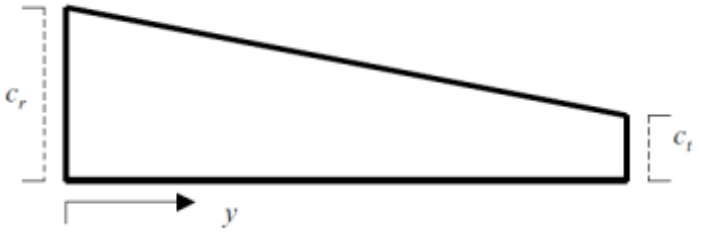


Figure 2.16. Schematic of half of a wing with nomenclature [9]

Therefore, to obtain these values, figures 2.17 and 2.18 have been used. Knowing the wingspan of each RPAS a conversion from meters to cm has been performed, in order to compute c_t and c_r.

For the Global Hawk case:

Figure 2.17. Global Hawk wingspan [19]

$$\lambda = \frac{c_t}{c_r} = \frac{0.695\ m}{2.085\ m} = \frac{1}{3}$$

For the Ikhana case:

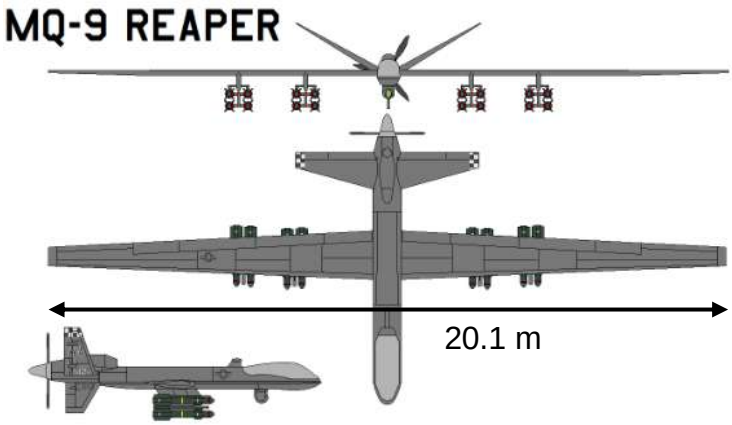


Figure 2.18. Ikhana wingspan [20]

$$\lambda = \frac{c_t}{c_r} = \frac{0.543m}{1.412\ m} = 0.384$$

Finally, RMC $_p$ can be computed and the value obtained can be substituted at equation (2.12) obtaining RMC $_{control.}$

The last step to know if the intensity of the wake vortex is still dangerous or not, a last calculation must be done applying equation (2.17)

$$r = \frac{RMC}{RMC_{control}} \quad \textbf{(2.17)}$$

Where,

r: is the ratio of intensity

RMC is the vortex rolling moment coefficient

$RMC_{control}$: is the RPAS control rolling moment coefficient

Depending on the value obtained from this ratio, it can be known if its hazard situation or not.

if { r > 1 ⟶ Hazard situation
{ r < 1 ⟶ Safe situation

Therefore, if the ratio is smaller than one the RPAS can continue its trajectory without worrying about the wake vortex, because the RMC $_{control}$ is larger than RMC. However, if it is larger than one the RPAS cannot go flying in that direction because if it does it, the vortex will turn it upside down. The value of RMC is larger than the one of control in this case and as a consequence the RPAS must change its trajectory in order to do not intercept with the wake vortex generated by the airliner.

2.3. Preliminary Results

As explained in section 2.1, the scenario used to assure the algorithm created works correctly is situating the airliner with a heading of 180º and make 360 iterations, one for each RPAS heading (figure 2.19). Therefore, the RPAS heading will go from 0º up to 359º. In the first iteration, the scenario will be the airliner flying with a heading of 180º and the RPAS with one of 0º, whereas in the last iteration the RPAS will be flying with a heading of 359º but the airliner will remain with the same heading. What is more, the airliner and the RPAS will be situated in a XY plane. The airliner with a position [5,5] NM and the RPAS depending on its heading will be in a different position of a circumference of radius 5 NM. The RPAS position is defined by equations (2.18) and (2.19).

$$x_{0_{RPAS}} = x_{0_{airliner}} - r \cdot \sin(\theta_{RPAS}) \quad \textbf{(2.18)}$$

$$y_{0_{RPAS}} = y_{0_{airliner}} - r \cdot \cos(\theta_{RPAS}) \quad \textbf{(2.19)}$$

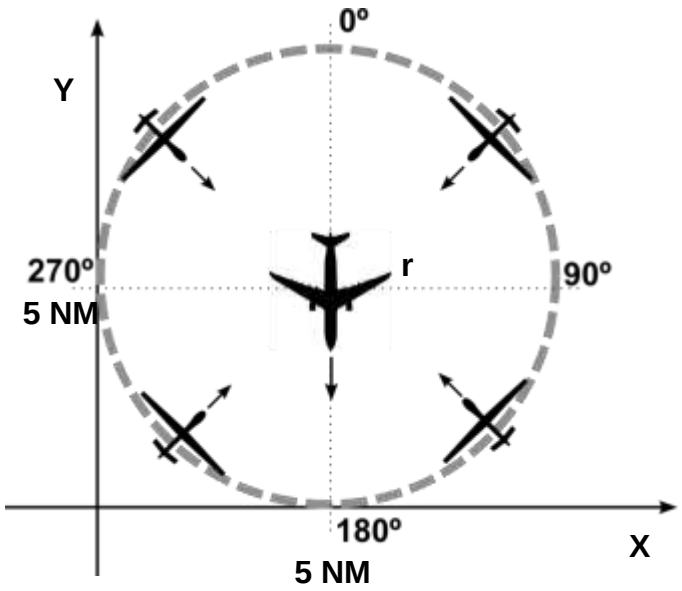


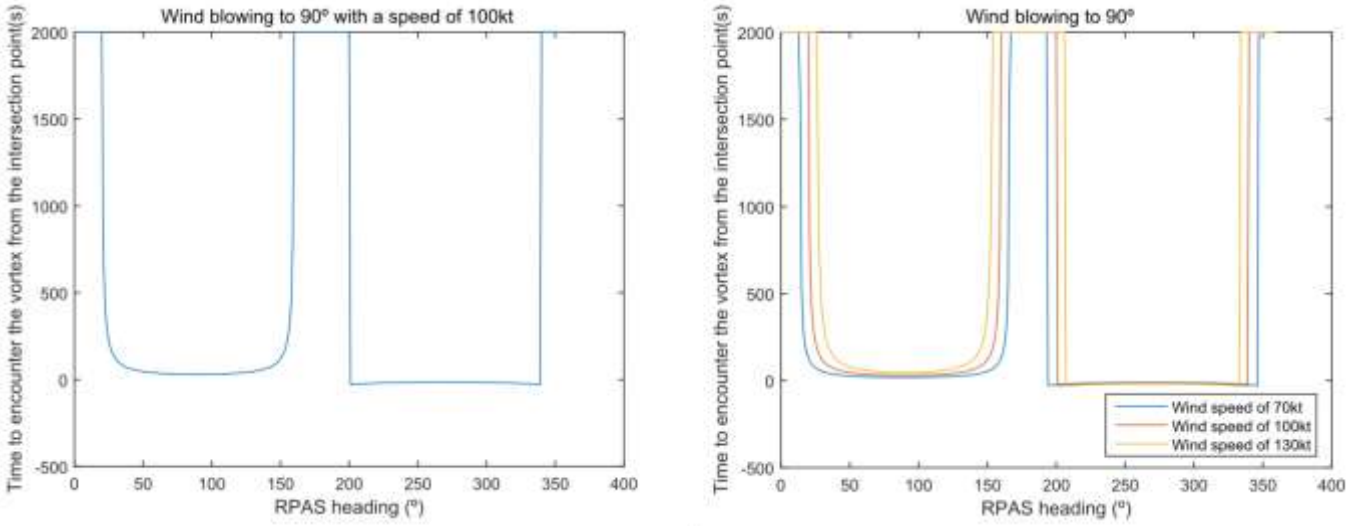
Figure 2.19. Scenario to check the algorithm

As the airliner is always flying with the same heading, it can be proved if the results obtained from the algorithm are correct for all the possible scenarios.

The results to be analysed are the time the RPAS will need to encounter the vortex from the intersection point between the airliner and RPAS trajectories versus the RPAS heading. Furthermore, there will be different results depending on the wind direction and speed.

In order to be able to plot when the time tends to infinite, it has been plotted a value of 2000 seconds in the graphic.

The first analysis is with a wind blowing to 90º with a speed of 100kt.The results obtained are the ones of figure 2.20 a) .



a) b)

Figure 2.20. RPAS hdg vs $t_{encounter.}$ a) Wind blowing to 90° with a speed of 100kt. b) Wind blowing to 90º with a speed of 70 kt, 100 kt and 130 kt

On the one hand, from figure 2.20 a) can be appreciated that when the RPAS heading is more similar to the airliner heading, so the RPAS trajectory and the vortex one are closer to be parallel, the time to reach the vortex tends to infinite, which has sense, because as more parallel two lines are, the more time it will be needed to reach the intersection between those lines. What is more, if the time to intersect the vortex is larger than 2000 seconds which corresponds to 33 minutes, in this model will not have more importance because the vortex cannot be active so much time and that is why all of them are considered 2000 seconds. Furthermore, as in this scenario the wind is blowing perpendicular to the airliner trajectory, this is the situation when the vortex will be more displaced from the airliner trajectory and when more RPAS headings will tend to infinite.

On the other hand, as the wind is blowing to 90º, if it is looked at figure 2.19 the vortex will be displaced to the right, for this reason what was expected to obtain was negative times for RPAS headings larger than 180º and smaller than 360º, because when the RPAS is at the intersection point it would have already passed through the airliner vortex, hence the time to encounter the vortex was a certain time in the past. Whereas, for RPAS headings larger than 0º and smaller than 180º, the expected time values were positive, because when the RPAS is

at the intersection point it will need some more time to reach the vortex. Both of the situations described are shown in figure 2.20 a), consequently it has been proved that the algorithm works correctly.

The RPAS headings from 21º up to 158º, the times to encounter the vortex are positive, whereas between 201º up to 339º the times are negative.

In the second analysis, the wind speed of the first scenario is changed. Two cases will be analysed, one with larger speed (130 kt) and another with a smaller one (70 kt). From the first case, larger speed, as the vortex will be more deviated, it is waited that there will be more RPAS heading tending to infinite, whereas in the second case, smaller speed, as the vortex will be less deviated than with a speed of 100 kt, the RPAS heading tending to infinite are expected to be less.

It can be clearly seen at figure 2.20 b) that increasing the wind speed maintaining the same wind angle makes more RPAS headings to tend to infinite, whereas there is the same tendency in which headings will be found negative or positive encounter times. Nevertheless, if it is observed the blue line which corresponds to 70kt, decreasing the wind speed and maintaining also the wind angle there are less headings that tend to infinite. Therefore what it was expected to obtain is what it has been got.

The last study will be changing from scenario one the angle where the wind is blowing to. As has been said in the first analysis, the maximum vortex deviation is when the wind is blowing perpendicular to the airliner trajectory. For that reason, if the wind blows in any different direction to the perpendiculars ones, with the same wind speed, the vortex deviation must be smaller and consequently less RPAS headings tending to infinite and that is exactly what in figure 2.21 is seen.

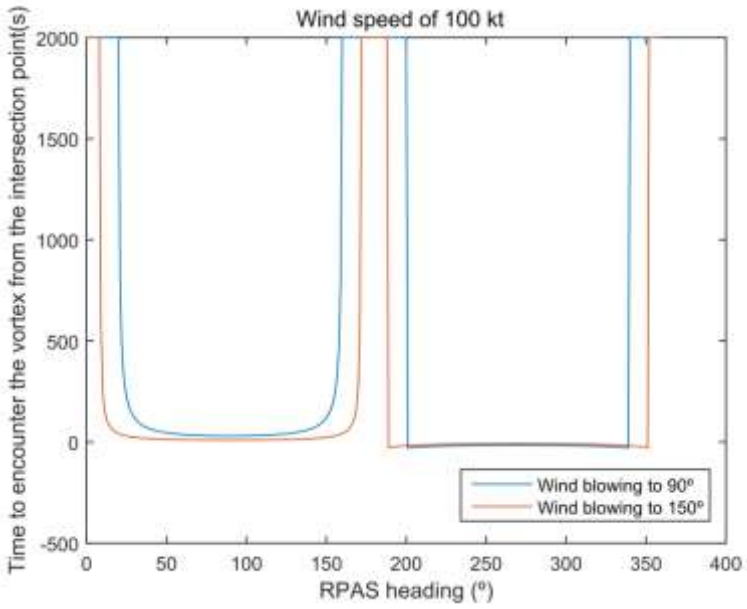
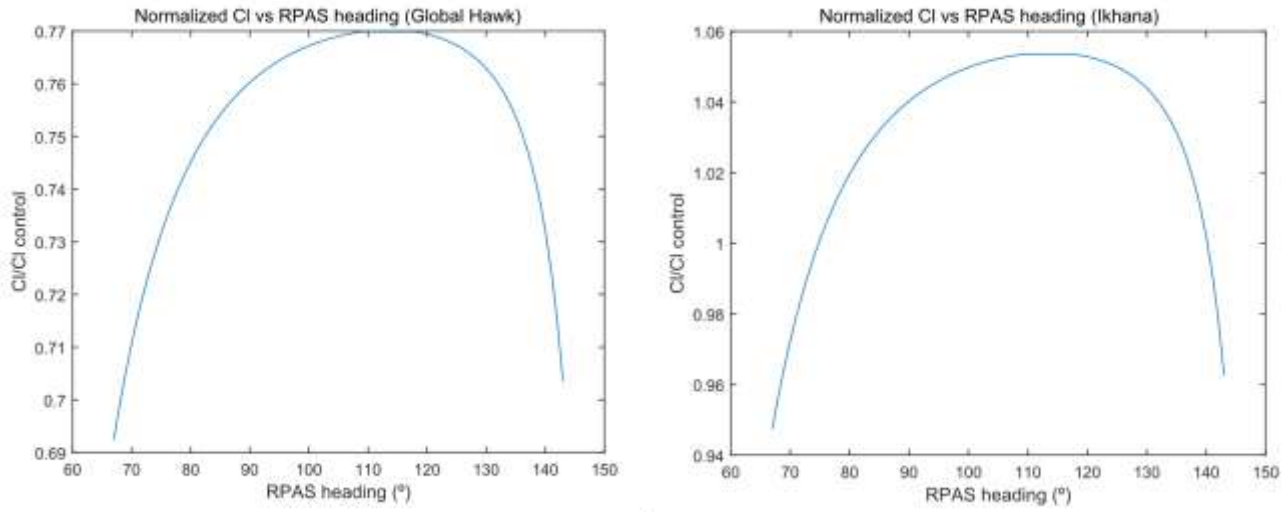


Figure 2.21. RPAS hdg vs $t_{encounter.}$ Wind blowing to 150° and 90° with a speed of 100 kt

Moreover, if the wind blows opposite to the airliner heading the wake vortex generated by the aircraft will not be deviated from the airliner trajectory. In this situation all the times the RPAS will need to encounter the vortex from the intersection point will be zero because the encounter point will be the intersection point.

From figure 2.20 and 2.21, the RPAS headings that does not tend to infinite are the ones that in a few minutes could encounter the vortex. Nevertheless, it is analysed if with the time needed to encounter the vortex the RPAS is inside the vertical and the horizontal vortex area. If it is inside, its intensity must be analysed to know if a hazard situation will be produced or not. Taking the same situation such as in figure 2.20 a), the RPAS headings which will bring the RPAS to encounter the vortex are the ones than can be observed in figures 2.22. At figure 2.22 a) the RPAS crossing the vortex is the Global hawk, whereas at b) is Ikhana.



a) b)

Figure 2.22. Normalized Cl vs RPAS heading. a) Global Hawk. b) Ikhana

On the one hand, from figure 2.22 a) can be appreciated that in any case there will not be a really dangerous situation that will make the RPAS turn upside down because none of the normalized rolling moment coefficient is larger than one. Therefore, the Global Hawk will only have turbulences if comes in these concretes headings.

On the other hand, from figure 2.22 b) can be observed that in some headings the normalized rolling moment coefficient is larger than one. In these situations the Ikhana cannot fly through the vortex because it will turn it upside down. It must change its trajectory before reaching it. In the cases where it is smaller than one, the control system of the RPAS will be able to compensate the rolling moment produced by the vortex.Therefore, a RPAS under the separation specified by the controller can be in a hazard as a consequence of the wake vortex, although the RPAS maintains the minimum safe separation of 5NM.

CHAPTER 3. TURN MODEL

In the past section only straight trajectories were considered. In this part, turns in the airliner manoeuvre will be introduced. Due to the turn introduction, this section will present the turn algorithm and also the modifications that must be applied to the collision algorithm in order to detect possible conflicts when the airliner is turning.

The turn algorithm will be based on the airliner heading change and bank angle. Moreover, in this analysis also the straight trajectory once the airliner has the heading desired will be considered. Finally, once the airliner position is known, the collision algorithm with the appropriate modifications in order to find the vortex encounter point will be applied.

3.1. Turn algorithm

As an assumption in this turning model, the airliner will always turn to the left once the airliner heading is 90º. Furthermore, the heading change and the bank angle the airliner will perform the turn will be inputs in this algorithm. The heading change is how many degrees has to change the airliner its heading to obtain a new one with this change.

Another significant factor to take into account is the time how long this situation is analysed. It is important because with this certain time, the airliner could have performed the heading change and once it has it, it cannot continue turning, it must go straight with this new heading. Therefore, a combination of turning and rectilinear movement will be combined if time is large enough.

To define the turn algorithm it must be defined the forces that will appear when a turn is performed. From the force diagram in figure 3.1 will be extracted the equations needed to obtain the airliner position after the turn.

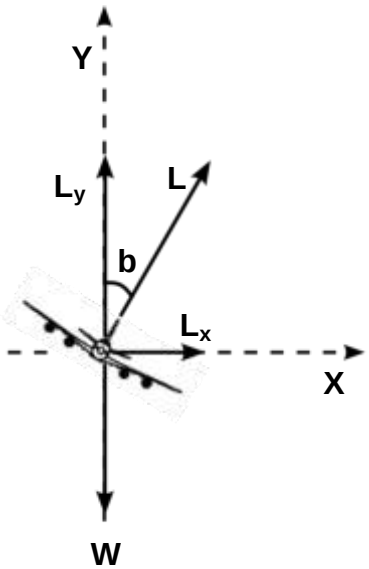


Figure 3.1. Force diagram

$$X \rightarrow L_x = L \cdot \sin(b) = m \cdot a_c \quad \textbf{(3.1)}$$

$$Y \rightarrow L_y = L \cdot \cos(b) = m \cdot g \quad \textbf{(3.2)}$$

Where:

L: Airliner lift [N]

m: Airliner mass [kg]

L_x : x lift component [N]

L_y : y lift component [N]

a_c : Centripetal acceleration [m/s^2]

b: Bank angle [º]

g: Gravity [m/s^2]

Equations (3.1) and (3.2) are divided obtaining equation (3.3).

$$\tan(b) = \frac{a_c}{g} = \frac{V^2}{R \cdot g} \quad \textbf{(3.3)}$$

Where:

V: Airliner speed [kt]

R: Turning radius [NM]

Isolating R, equation (3.4) is obtained.

$$R = \frac{V^2}{g \cdot \tan(b)} \quad \textbf{(3.4)}$$

Once the turning radius is obtained, the angle the airliner will turn must be also known. This angle must be larger than $\frac{3\pi}{2}$, because it is the angle at which the turn will begin in order to turn to the left, as shown in figure 3.2.

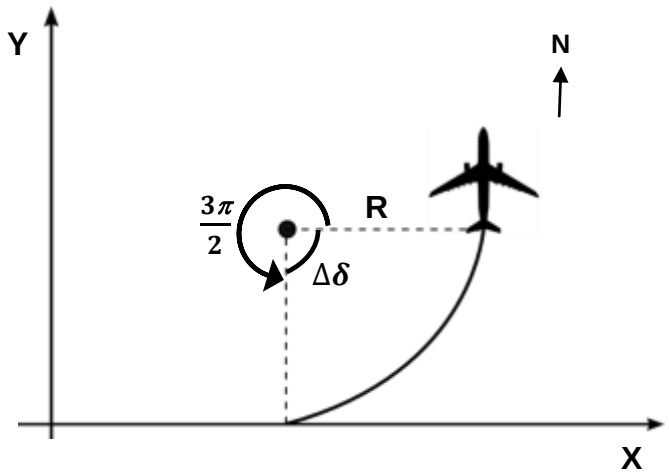


Figure 3.2. Definition of the turn angle

Consequently, the turn angle will be:

$$\varphi = \frac{3\pi}{2} + \Delta\delta \quad \textbf{(3.5)}$$

Where,

φ: Turn angle [rad]

$\Delta\delta$: Heading variation [rad]

With all this data the new position of the airliner after the turn is obtained by equations (3.5) and (3.6).

$$x_{airliner} = x_0 + R \cdot \cos(\varphi) \quad \textbf{(3.5)}$$

$$y_{airliner} = y_0 + R + R \cdot \sin(\varphi) \quad \textbf{(3.6)}$$

Nevertheless, the position obtained in equations (3.5) and (3.6) is where the airliner will be if the time analysed is smaller than the time the airliner needs to perform the turn and obtain the heading desired, which would mean the airliner is still turning, such as in figure 3.3.

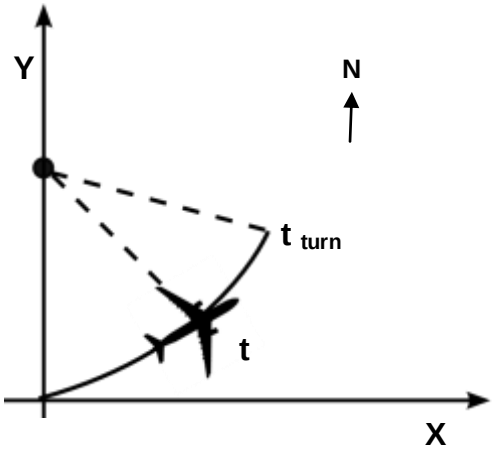


Figure 3.3. Aircraft still performing the turn ($t<t_{turn}$)

However, if the time analysed is larger than the time the airliner needs to end the turn and get the new heading, the airliner would have end the turn and the difference of these times will be the time it flies performing a rectilinear movement and not a turn, as can be observed in figure 3.4.

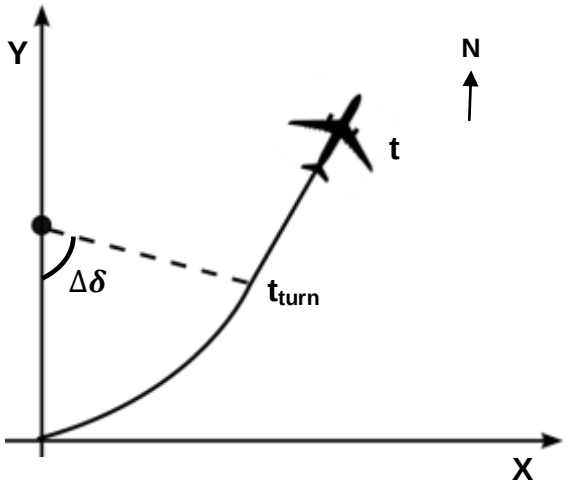


Figure 3.4. Airliner flying with constant heading ($t>t_{turn}$)

For that reason, in this second situation some equations must be added. From equation (3.5) and (3.6) is obtained the position where the airliner will be when the turn ends. To this value must be added the distance the airliner will travel with rectilinear movement, getting the final position where it will be after this time analysed. Equations (3.7) and (3.8) are used to obtain this final position.

$$x_{airliner} = x_{airliner\ turn} + l \cdot \sin(\theta_t) \quad \textbf{(3.7)}$$

$$y_{airliner} = y_{airliner\ turn} + l \cdot \cos(\theta_t) \quad \textbf{(3.8)}$$

Where:

$x_{airliner\ turn}$: $x_{airliner}$ from equation (3.5) [NM]

$y_{airliner\ turn}$: $y_{airliner}$ from equation (3.6) [NM]

l: distanced to be travelled with a rectilinear movement [NM]

θ_t: Airliner heading after the turn [º]

The new airliner heading after the turn is acquired applying equation (3.9), and this formula is based on figure 3.5.

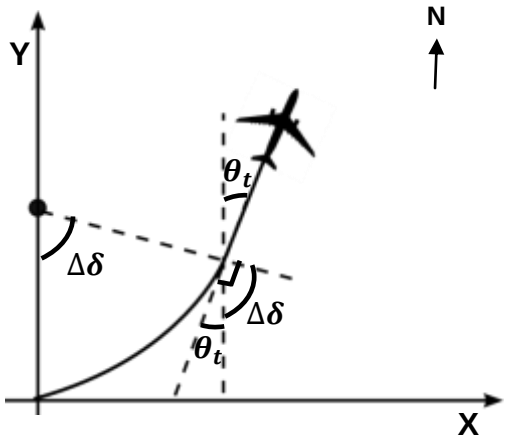


Figure 3.5. Deduction of $\boldsymbol{\theta_t}$

$$\theta_t = 90^\circ - \Delta\delta \quad \textbf{(3.9)}$$

Once the final new position is obtained, it is returned to the collision algorithm to compute the time will need the RPAS to get the encounter point with the vortex.

3.2. Collision algorithm modifications

With the implementation of the airliner's turns, the collision algorithm created in the section 2 must have some modifications in order to be able to compute the time the RPAS will need to encounter the vortex.

The first difference is that to know the position where the airliner will be when the RPAS is at the intersection point between trajectories, it is no longer used a rectilinear movement if there is the condition the airliner is performing a turn. It is used the turn algorithm which will return the airliner position and its new heading, taking into account how long the airliner has flown over that point.

Once this new situation is known, the axis are moved, making the origin be now the airliner position. As can be seen in figure 3.6, the airliner will be at [0,0] and the RPAS will be at RPAS position minus the airliner position after the turn.

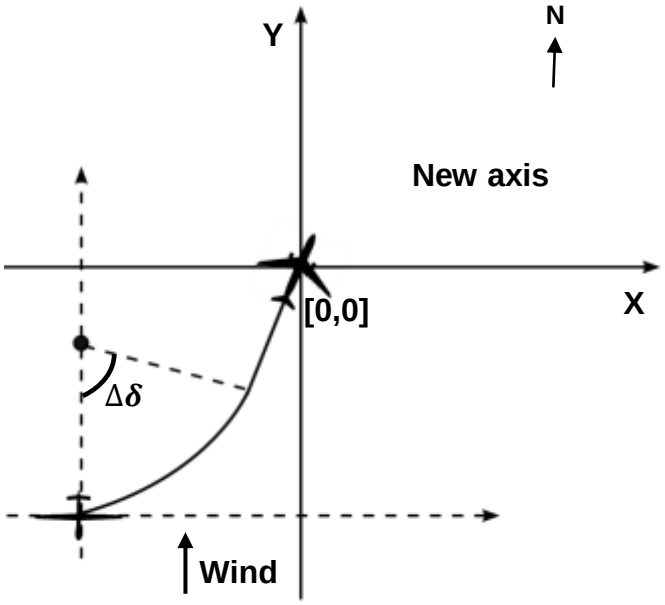


Figure 3.6. New axis position

In addition, as after the turn the airliner will not have a heading of 90º, once the axis are moved it is performed a rotation to get the airliner heading be again 90º. The airliner position will be the same, the origin of coordinates, but the RPAS position will have changed after this rotation. Therefore, the rotation

matrix will be used and the new situation of the RPAS will be the one obtained by equation (3.10).

$$\begin{pmatrix} x_{RPAS} \\ y_{RPAS} \end{pmatrix} = \begin{pmatrix} \cos(\sigma) & \sin(\sigma) \\ -\sin(\sigma) & \cos(\sigma) \end{pmatrix} \begin{pmatrix} x_{0,RPAS} \\ y_{0,RPAS} \end{pmatrix} \quad \textbf{(3.10)}$$

Where:

$x_{0,RPAS}$: RPAS x position before the rotation [NM]

$y_{0,RPAS}$: RPAS y position before the rotation [NM]

σ: Angle of rotation [º]

Therefore, after the rotation the scenario that is going to be analysed is the one of figure 3.7.

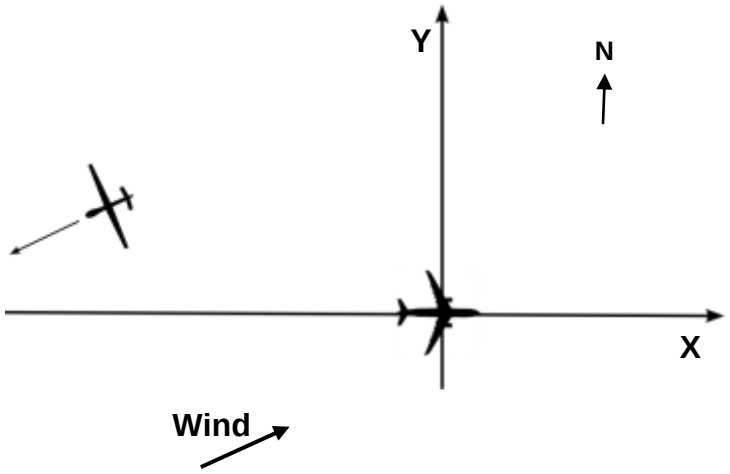


Figure 3.7. Scenario after the rotation

With this situation it can be computed the vortex deviation angle (α) in the same way as if the airliner has not turned. Nevertheless, if a turn has been performed, the way of computing the time the RPAS will need to encounter the vortex is not the same. What it is done in this case is defining the vortex and RPAS trajectories, obtaining one equation for each straight line. Then both equations are equated and the encounter *x* and *y* positions of figure 3.8 are obtained. Next, the time to encounter this point is computed, but the way to define the sign of this time depending on if the RPAS will encounter this point in the future, such as in figure 3.8, (positive sign) or it has already passed in a certain time in the past (negative sign), is different than in the no turn case. In this situation, it will not depend on to where is blowing the wind, but instead it will depend on the RPAS heading and if its *y* position is larger or not the encounter one.

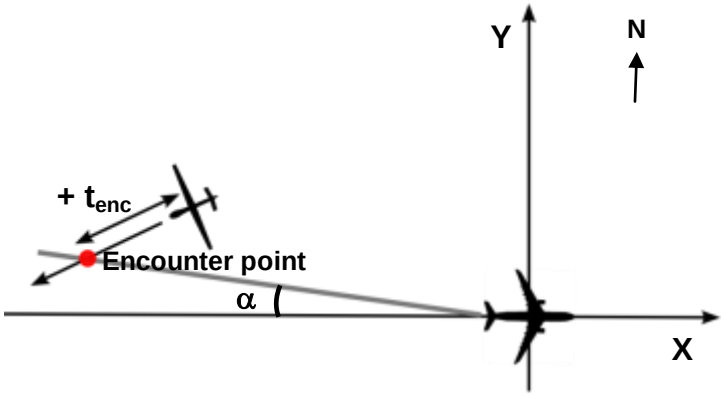


Figure 3.8. Scenario to compute the encounter time

However, this first encounter time found, as in the other situation of no turn, will not be the real one, therefore the loop explained in section 2 must be applied. What is more, in the turning situation the mechanism of searching this time is exactly the same than in the other case. What changes is that the airliner position will not be computed as a rectilinear movement. It will be used the turn algorithm to know its position that could be at the turn or already flying with a constant heading. Then, it will be done the same as explained above: changing the position of the axis, rotating until the airliner heading is 90º, computing the new RPAS position, finding the new encounter point and finally computing the new time encounter.

Once the encounter time is found, the vertical and horizontal vortex analysis are exactly the same.

3.3. Preliminary Results

In this section, the results obtained to analyse will be extracted from the same scenario than in section 2.3.The airliner will go with the same heading in all the iterations, but this time it will also turn after the intersection point between trajectories. In addition, in this model where the airliner is able to turn, not only the wind direction and speed are parameters that can change, also the bank angle and the variation of heading can be chosen. Such as the effect of the wind was well analysed in section 2.3, in this part will be studied how the bank angle and the variation of heading affect to the time the RPAS will need to encounter the wake vortex generated by the airliner.

A simulation that can check if the turning model works correctly is assigning the variation of heading equal to zero. With this assignment the airliner will not perform any turn, therefore the graphic obtained is the same as figure 3.3.

The first analysis will consist on fixing a heading variation, for instance 20º, and changing the bank angle. In that way, it will be able to see the behaviour of the

time the RPAS will need to get the vortex depending on the bank angle the airliner is performing the turn.

In figures 3.9 and 3.10, the bank angle is equal to 30º and 20º, respectively. However, the heading variation is in both cases 20º. As can be observed, the number of headings that tend to infinity in both figures is the same. That happens due to in both cases there is the same heading variation, making the airliner after each RPAS heading analysis, to have the same heading in both scenarios. Therefore, as the wind direction and intensity in both cases is also the same, the RPAS headings that will need an infinite time to encounter the vortex will be the same in both situations. Nevertheless, as a consequence of having a different bank angle, what changes a little bit is the time the RPAS needs to reach the vortex when does not tend to infinity. In the first case, where the bank angle is equal to 30º, the RPAS will need in some cases more time to reach it, whereas in the second case, where the bank angle is equal to 20º, it will need a little less time to encounter the wake vortex, but in some other cases both will need the same amount of time.

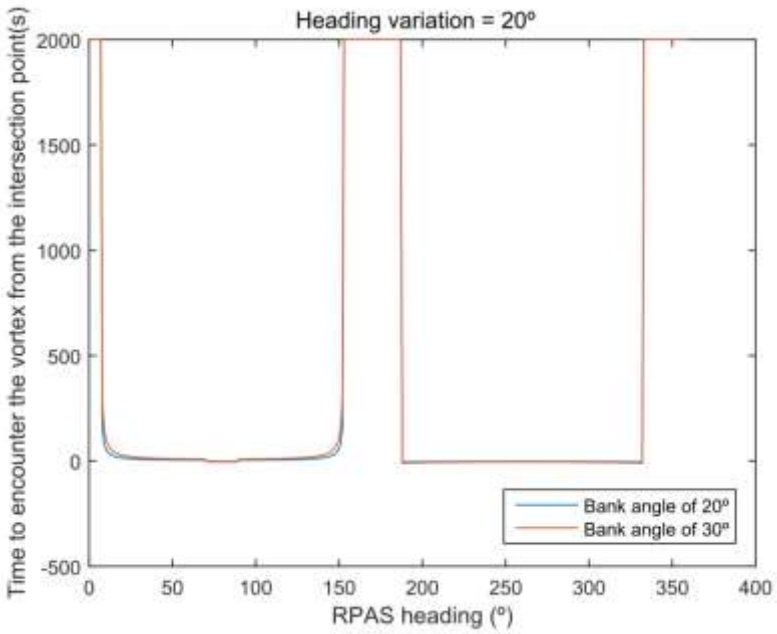


Figure 3.9. Bank angle of 20° and 30°, heading variation of 20°

The second analysis will consist on changing the heading variation and maintaining the same bank angle, for instance 20º.

What can be extracted observing figure 3.10 is that maintaining the same bank angle but changing the heading variation, there are more angles that tend to infinite with a larger heading variation. Another aspect that changes is the time the RPAS will need to get the vortex depending on its heading. In the case where the heading variation is 30º, the time where in the case of 20º of heading variation where positive, now are lightly negative, and the ones that where negative are lightly positive. This situation is as a consequence of choosing a

different new heading for the airliner, making a total change situation between both cases, which ends with the RPAS needing different time for the same heading.

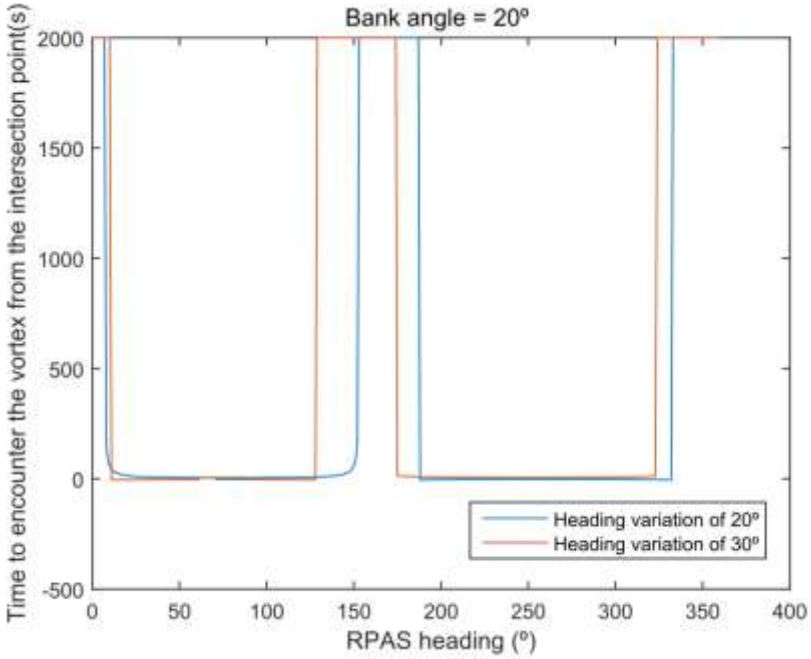


Figure 3.10. Heading variation of 20° and 30°, bank angle of 20°

The third analysis will consist on study the vortex intensity and see in which of the possible RPAS headings of figures 3.9 and 3.10 there will be a hazard situation. First, the vertical and horizontal analysis of the vortex extension are performed and the RPAS headings in which the RPAS will encounter the wake vortex are the ones of figure 3.11.

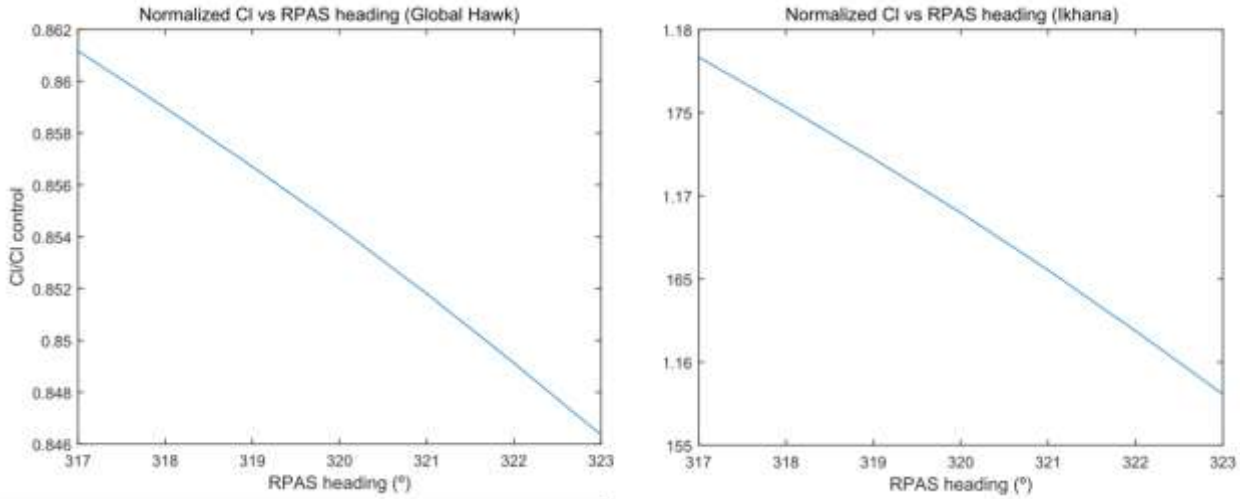


Figure 3.11. Normalized Cl vs RPAS heading. a) Global Hawk. b) Ikhana

In both cases, the headings in which the vortex will affect both RPAS are the same, but what changes is the normalized rolling moment coefficient. If the RPAS is the Global Hawk (figure 3.11 a)), it could perform its trajectory because the vortex does not have enough intensity to make it turn upside down. However, it will experience some turbulences. Whereas, in the Ikhana case (figure 3.11 b)), it must change its trajectory because in all the cases, when it encounters the vortex, the vortex will be able to turn it upside down.

Conclusions

The goal of this project was to obtain a system that detects if there will be a collision between a RPAS and the wake vortex generated by a commercial aircraft. This achievement was pursued because the introduction of RPAS in civil airspace with their different performances comparing with airliners, make them not to be included in the same classification such as the airliners for separation terms. Their wingspans are much larger compared with the commercial aircraft, but their weight are much lower. Therefore, the wake vortex categorisation in order to perform a safe separation between RPAS and airliners is not valid. Consequently, a system that allows to know if there will be a hazard situation with the wake vortex is necessary.

To do that, a simple theoretical wake vortex model has been created in order to implement the collision models. In this model, the atmospheric conditions at which the vortex is wanted to be analysed can be chosen. The generator can also be chosen, which is the commercial aircraft, introducing its weight and dimensions. Nevertheless, at the simulation performed to know if the model works correctly, only the A320 has been used. Next, two collision models have been created, one for straight trajectories and the other for taking also into account the turns the airliner can perform. In both cases, a vertical and a horizontal analysis of the vortex can be found. First the vertical one is executed and if the RPAS is inside the vertical vortex area, then the horizontal analysis is carried out. If the RPAS is also in the horizontal area, then the intensity of the vortex in this point is analysed. The strength of the wake vortex can be quantified non-dimensionally by calculating the rolling moment coefficient, but to obtain it, the value of circulation must be also computed and it is based on the atmospheric conditions chosen.

At the end of both collision models, if the RPAS is inside the vortex area and the value of the rolling moment coefficient divided by the roll supplied by the airplane control system (normalized rolling moment) is larger than one, that means the RPAS is not able to compensate the roll produced by the airliner wake vortex and it will have an accident because the vortex can turn it upside down. In addition, the preliminary results obtained for instance at the collision model without taking into account the turns, situates the RPAS at a distance of 5 NM from teh airliner and at this moment it is analysed if it will have problems with the vortex or not. There are some RPAS headings that will have a hazard situation. Therefore, this preliminary simulation can be extracted to a real case because the distance maintained by the RPAS with the airliner is one that a controller can assign to the RPAS, and with this separation it can be in hazard situation due to the wake vortex of the commercial airliner, although it is in a supposed safe area. Nevertheless, if the normalized rolling moment coefficient

is smaller than one, the RPAS could experience turbulences, but it will be capable to compensate the rolling moment generated by the wake vortex and continue its way without serious problems.

To sum up, if the vortex conflict detection system gives a normalized rolling moment larger than one, the RPAS cannot continue with its trajectory if it wants to perform a safe mission.

Overall, wake vortices are a serious hazard for flight at altitude and not only at departures and take off procedures at airports. The danger is especially prevalent for RPAS and also for smaller regional jet aircraft, which in some scenarios might not be able to create enough control power to overcome the effects of the wake vortex generated by larger commercial aircrafts.

To conclude, this project is a simple theoretic model that can be used in further invesigations where real scenarios can use the collision models proposed in orther to prevent the RPAS having an accident.This is because when it was simulated to check if the collision model worked well, it was using the minimum safe separation and with this situation that it is assumed the RPAS to be in a safe location, actually it is not. It is in a hazard situation where the vortex can turn it upside down. Therefore, further studies must center its objective in simulating real scenarios introducing the RPAS in the air traffic using this detection vortex system for RPAS operations in order to do not have an accident when it is supposed to be in a safe position.

Bibliography

[1] Aeronautical Information Circular P 001/2015, United Kingdom. NATS Services, January 2015.

[2] Advisory circular. "Aircraft Wake Turbulence". U.S. Department of Transportation, 2014, AC Nº: 90-23G

[3] Flight Operations Briefing Notes. Operating Environment. Wake Turbulence Awareness/Avoidance, Airbus.

[4] RECAT - EU. "European Wake Turbulence Categorisation and Separation Minima on Approach and Departure", 2015, Edition Number: 1.0

[5] IVAO, "Wake Turbulence Separation Minima", Version 1.0, 2014.

[6] Borivoj Galovic, Berislav Grozdanic, Sanja Steiner, "Influence of Wake-Vortex Turbulence on the Flight Safety". Vol. 9, 1997, No. 1-2, 1-14.

[7] Ivan De Visscher, Grégoire Winckelmans and Vincent Treve. "A Simple Wake Vortex Encounter Severity Metric". Eleventh USA/Europe Air Traffic Management Research and Development Seminar, ATM 2015.

[8] Fred H.Proctor and Nash' at N. Ahmad, George S. Switzer, Fanny M. Limon Duparcmeur. "Three-Phased Wake Vortex Decay". Virginia, 23681, 23666.

[9] T.Economon, "Effects of Wake Vortices on Commercial Aircraft". *AIAA 2008-1428*. University of Notre Dame, Notre Dame, Indiana, 46556.

[10] Mike Hoogstraten and Hendrikus G. Visser, Dennis Hart, Vincent Treve and Frederic Rooseleer. "Improved Understanding of En Route Wake-Vortex Encounters", *Journal of Aircraft*, Vol.52, No.3, May - June 2015.

[11] Chao Liu. "Wake Vortex Encounter Analysis with Different Wake Vortex Models Using Vortex-Lattice. A numerical study" . November 2007.

[12] Fred H.Proctor and David W. Hamilton. "Evaluation of Fast-Time Wake Vortex Prediction Models". NASA Langley Research Center, Hampton, Virginia, 23681.

[13] Holzäpfel, F. and Robins, R.E., "Probabilistic Two-Phase Aircraft Wake-Vortex Model: Application and Assessment," *Journal of Aircraft*, Vol.41, No.5, 2004, pp. 1117-1126.

[14] Holzäpfel, F., "Probabilistic Two-Phase Aircraft Wake-Vortex Model: Further Development and Assessment, "*Journal of Aircraft*, Vol.43, No. 3, 2006, pp. 700-708.

[15] Holzäpfel, F., "Probabilistic Two-Phase Wake-Vortex Decay and Transport Model," *Journal of Aircraft*, Vol. 40, No. 2, March-April, 2003, pp. 323-331.

[16] Proctor, F.H., and Switzer, G.F., "Numerical Simulation of Aircraft Trailing Vortices," 9th Conf. on Aviation, Range and Aerospace Meteorology, American Meteorology Society, 11-15 Sept 2000, pp. 511-516.

[17] Switzer, G.F., and Proctor, F.H., "Numerical Study of Wake Vortex Behaviour in Turbulent Domains with Ambient Stratification, "38th Aerospace Sciences Meeting & Exhibit, 10-13 January 2000, AIAA-2000-0755.

[18]http://www.airbus.com/aircraftfamilies/passengeraircraft/a320family/a320/specifications/ [Consulted 12/03/2016]

[19] Northrop Grumman Integrated Systems. "Global Hawk Integrated Risk Management". 26 October 2006.

[20] Peter W. Merlin. "Ikhana. Unmanned Aircraft System, Western States Fire Missions" Monographs in Aerospace History #44. NASA SP-2009-4544

APPENDICES

TFG TITLE: Enroute vortex conflict detection system for RPAS operations

DEGREE: Grau en Enginyeria d'aeronavegació

AUTHOR: Mònica Marcos Benítez

ADVISOR: Marc Pérez Batlle

DATE: 18 july, 2016

Appendix A. Matlab code. Collision model

Main

```
close all;
d=5;%NM
i=1;
phy=0;
t1=[0 9 18 27 39 57 72 78 84 93 102 105 108 111 114 117 120 123 126
129 138 147 156 165];
c1=[575 550 540 520 500 480 460 450 440 430 420 410 400 390 360 340
320 300 280 260 200 150 100 70];
t2=[0 3 9 12 21 24 27 30 33 39 45 48 57 63 75 84 96 105];
c2=[575 560 550 540 530 520 510 500 480 440 400 360 300 260 200 160
100 70];
t3=[0 9 12 21 30 39 51 57 66 72 78 81 84 87 90 93 96 102 105 114 120];
c3=[575 550 540 530 510 500 490 480 470 460 440 420 400 380 320 300
260 220 190 100 70];
atmosphere1=true; %The type of atmospheric conditions is chosen
atmosphere2=false;
atmosphere3=false;
global_hawk=true; %The type of RPAS used for the simulation is chosen
ikhana=false;
m=1;

while(phy<360)
    angle(i)=phy;
    hdg_RPAS=phy;
    t=1;
    %Airliner data is introduced
    airliner_span=35.8;% airliner wingspan in meters
    airliner_mass=64500; %airliner mass 64500 in kg (Max landing
weight A320)
    X0_airliner=5;%Initial x position (NM)
    Y0_airliner=5;%Initial y position (NM)
    V_airliner=500;%Airliner speed (knots)
    h_airliner=35000;%Airliner altitud (ft)
    hdg_airliner=180;%Airliner heading (°)
    %Wind data is introduced
    beta=90;%Direction to where the wind is blowing (°)
    w=100;%Wind speed(knots)
    vert_wind_V=0; %Vertical wind speed (wind speed in the z
direction) (ft/min). Positive if it goes downwards and negative if it
goes upwards.
    % The x and y speed components are computed
    u_airliner=V_airliner*sind(hdg_airliner);%Airliner velocity in the
x direction (knots)
    v_airliner=V_airliner*cosd(hdg_airliner);%Airliner velocity in the
y direction (knots)
    %A new point is created to make the airliner line equation
    X1_airliner=X0_airliner+u_airliner*t;
    Y1_airliner=Y0_airliner+v_airliner*t;
    %Vector of the line is created
    vect_1_airliner=X1_airliner-X0_airliner;
    vect_2_airliner=Y1_airliner-Y0_airliner;
    %Airliner equation line
    %Y_airliner=((X_airliner-
X0_airliner)/vect_1_airliner)*vect_2_airliner+Y0_airliner;
    if(hdg_RPAS==hdg_airliner||abs(hdg_airliner-hdg_RPAS)==180 )
```

```
        hdg_RPAS=hdg_RPAS+1;
    end
    %RPAS data is introduced
    if(global_hawk==true)
        AR_RPAS=25;
        RPAS_span=39.9;%RPAS wingspan in meters (global hawk)
    else if(ikhana==true)
            AR_RPAS=17;
            RPAS_span=20.1;%RPAS wingspan in meters (Ikhana)
        end
    end

    X0_RPAS=-d*sind(hdg_RPAS)+X0_airliner;%Initial RPAS x position
(NM)
    Y0_RPAS=-d*cosd(hdg_RPAS)+Y0_airliner;%Initial RPAS y position
(NM)
    V_RPAS=300;%RPAS speed (knots)
    h_RPAS=34200;%RPAS altitude (ft)

    %The velocity x and y components are computed
    u_RPAS=V_RPAS*sind(hdg_RPAS);% RPAS velocity in the x direction
(knots)
    v_RPAS=V_RPAS*cosd(hdg_RPAS);%RPAS velocity in the y direction
(knots)
    %A new point is created to make the RPAS line equation
    X1_RPAS=X0_RPAS+u_RPAS*t;
    Y1_RPAS=Y0_RPAS+v_RPAS*t;
    %Vector of the line is created
    vect_1_RPAS=X1_RPAS-X0_RPAS;
    vect_2_RPAS=Y1_RPAS-Y0_RPAS;
    %RPAS line equation
    %Y_uav=((X_uav-Xo_uav)/vect_1_uav)*vect_2_uav+Y0_uav;

    %If the RPAS and the airliner fly at the same altitude  with
opposite headings, they will have a front collision
    if(abs(hdg_airliner-hdg_RPAS)==180 && h_RPAS==h_airliner)
        disp('A frontal collision will be produced')
    else
        %The two line equations are equated and the x intersection is
        %obtained
        X_intersec=(-
X0_airliner*vect_2_airliner*vect_1_RPAS+vect_1_airliner*Y0_airliner*ve
ct_1_RPAS+vect_1_airliner*X0_RPAS*vect_2_RPAS-
vect_1_airliner*vect_1_RPAS*Y0_RPAS)/(vect_1_airliner*vect_2_RPAS-
vect_2_airliner*vect_1_RPAS);

        %Substituing the x value of the intersection to one of the two
equations, the y value of the
        %intersection is obtained
        Y_intersec=((X_intersec-
X0_RPAS)/vect_1_RPAS)*vect_2_RPAS+Y0_RPAS;
        %To be able to compute the time the RPAS needs to get the
        %intersection point, first it must known the distance that
there
        %is. As the initial point where the RPAS and the intersection
point
        %are known the distance can be calculated.

        d_RPAS_intersec=((X_intersec-X0_RPAS)^2+(Y_intersec-
Y0_RPAS)^2)^(1/2);
```

```
        %Knowing the distance travelled and the speed at which it
flies, the time can be known
        t_RPAS_intersec=d_RPAS_intersec/V_RPAS;

        %To know the time the airliner needs to get to the
intersection point,
        %first the distance it will travel until this point is
computed
        d_airliner_intersec=((X_intersec-X0_airliner)^2+(Y_intersec-
Y0_airliner)^2)^(1/2);
        %With this distance and the speed at which it flies, the time
the airliner needs to get
        %to the intersection point is calculated
        t_airliner_intersec=d_airliner_intersec/V_airliner;

        %The airliner position is computed with the time the RPAS
needs to get
        %to the intersection point. To obtain the new y position, the
y
        %velocity is multiplied by time and the same is done to get
the new x
        %position

Y_airliner_RPAS_intersec=Y0_airliner+v_airliner*t_RPAS_intersec;

X_airliner_RPAS_intersec=X0_airliner+u_airliner*t_RPAS_intersec;
        t_airliner_after_inter=t_RPAS_intersec-t_airliner_intersec;
        %It is analysed if the RPAS is before or after the airliner,
using
        %diferent conditions depending on the airliner direction
        if(hdg_airliner>0 && hdg_airliner<180)
            if(X_airliner_RPAS_intersec>=X_intersec)
                disp('There can be a conflict with the vortex');

[t_enc(i),normalized]=detectConflictWind(ikhana,global_hawk,AR_RPAS,at
mosphere1,atmosphere2,atmosphere3,RPAS_span,airliner_mass,airliner_spa
n,vert_wind_V,h_airliner,h_RPAS,hdg_airliner,hdg_RPAS,beta,X_airliner_
RPAS_intersec,Y_airliner_RPAS_intersec,V_airliner,X_intersec,Y_interse
c,w,V_RPAS,t1,c1,t2,c2,t3,c3);
                if(normalized~=0)
                    vect_norm(m)=normalized;
                    vect_angle(m)=hdg_RPAS;
                    m=m+1;
                end


            else
                disp('There is no conflict with the vortex');
            end

        elseif(hdg_airliner>180 && hdg_airliner<360)
            if(X_airliner_RPAS_intersec<=X_intersec)
                disp('There can be a conflict with the vortex');

[t_enc(i),normalized]=detectConflictWind(ikhana,global_hawk,AR_RPAS,at
mosphere1,atmosphere2,atmosphere3,RPAS_span,airliner_mass,airliner_spa
n,vert_wind_V,h_airliner,h_RPAS,hdg_airliner,hdg_RPAS,beta,X_airliner_
RPAS_intersec,Y_airliner_RPAS_intersec,V_airliner,X_intersec,Y_interse
c,w,V_RPAS,t1,c1,t2,c2,t3,c3);
                if(normalized~=0)
```

```
                    vect_norm(m)=normalized;
                    vect_angle(m)=hdg_RPAS;
                    m=m+1;
                end
            else
                disp('There is no conflict with the vortex');
            end
        elseif(hdg_airliner==0)
            if(Y_airliner_RPAS_intersec>=Y_intersec)
                disp('There can be a conflict with the vortex');

[t_enc(i),normalized]=detectConflictWindairliner_mass(ikhana,global_ha
wk,AR_RPAS,atmosphere1,atmosphere2,atmosphere3,RPAS_span,airliner_mass
,airliner_span,vert_wind_V,h_airliner,h_RPAS,hdg_airliner,hdg_RPAS,bet
a,X_airliner_RPAS_intersec,Y_airliner_RPAS_intersec,V_airliner,X_inter
sec,Y_intersec,w,V_RPAS,t1,c1,t2,c2,t3,c3);
                if(normalized~=0)
                    vect_norm(m)=normalized;
                    vect_angle(m)=hdg_RPAS;
                    m=m+1;
                end
            else
                disp('There is no conflict with the vortex');
            end
         elseif(hdg_airliner==180)
            if(Y_airliner_RPAS_intersec<=Y_intersec)
                disp('There can be a conflict with the vortex');

[t_enc(i),normalized]=detectConflictWind(ikhana,global_hawk,AR_RPAS,at
mosphere1,atmosphere2,atmosphere3,RPAS_span,airliner_mass,airliner_spa
n,vert_wind_V,h_airliner,h_RPAS,hdg_airliner,hdg_RPAS,beta,X_airliner_
RPAS_intersec,Y_airliner_RPAS_intersec,V_airliner,X_intersec,Y_interse
c,w,V_RPAS, t1,c1,t2,c2,t3,c3);
               if(normalized~=0)
                    vect_norm(m)=normalized;
                    vect_angle(m)=hdg_RPAS;
                    m=m+1;
                end
            else
                disp('There is no conflict with the vortex');
            end


        end

    end
    i=i+1;
    phy=phy+1;

end
figure(1)
plot(angle,t_enc);
hold on;
title('Wind speed of 100 kt');
xlabel('RPAS heading (°)');
ylabel('Time to encounter the vortex from the intersection point(s)');

figure(2)
plot(vect_angle,vect_norm);
hold on
```

```
title('Normalized Cl vs RPAS heading (Global Hawk)');
xlabel('RPAS heading (°)');
ylabel('Cl/Cl control');
```

Function: DetectConflictWind

```
function[t_enc,normalized]=detectConflictWind(ikhana,global_hawk,AR_RP
AS,atmosphere1,atmosphere2,atmosphere3,RPAS_span,airliner_mass,airline
r_span,vert_wind_V,h_airliner,h_RPAS,hdg_airliner,hdg_uav,beta,X_airli
ner_RPAS_intersec,Y_airliner_RPAS_intersec,V_airliner,X_intersec,Y_int
ersec,w,V_RPAS, t1, c1,t2,c2,t3,c3)
%%Initialize variables
%Intersection of trajectories
intersection = [X_intersec,Y_intersec];
%RPAS position
RPAS_pos = [0,0];
%RPAS velocity
RPAS_speed=V_RPAS;
%Airliner position
airliner_pos=0;
%Airliner speed
airliner_spe=V_airliner;
%Wind
wind_speed=w;
%wind angle
wind_angle=beta;
%RPAS heading
RPAS_hea=hdg_uav;
%airliner heading
airliner_hea = hdg_airliner;
enc_x=0;
enc_y=0;
normalized=0;
%the airliners always flies to the east, so the airliner and the RPAS
are
%rotated in order to make the airliner fly with a heading of 90°
degrees.
angle_rotation=90-airliner_hea;
new_airliner_hea=airliner_hea+angle_rotation;%It must be 90°
new_RPAS_hea=RPAS_hea+ angle_rotation;
if(new_RPAS_hea<0)
    new_RPAS_hea=360+new_RPAS_hea;
end
new_wind_angle=wind_angle+angle_rotation;
if(new_wind_angle<0)
    new_wind_angle=360+new_wind_angle;
end
wind_spe = [wind_speed * sind(new_wind_angle), wind_speed *
cosd(new_wind_angle)];
RPAS_spe = [RPAS_speed * sind(new_RPAS_hea), RPAS_speed *
cosd(new_RPAS_hea)];

%calculate vortex angle
alpha = atand(wind_spe(2)/(wind_spe(1) + airliner_spe(1)));
%calculate angle of the conflict
beta = 90 - new_RPAS_hea;

%At t=0 the RPAS is in the intersection point. Then, situate airliner
%following the initial conditions
```

```
airliner_pos(1) = airliner_pos + abs(((X_airliner_RPAS_intersec-
X_intersec)^2+(Y_airliner_RPAS_intersec-Y_intersec)^2)^(1/2));

%find encounter angle
gamma = 180 - beta - alpha;

%find vortex encounter position
enc_y_prove= airliner_pos * tand(alpha);
enc_x_prove= enc_y_prove/tand(beta);
enc_prove = [enc_x_prove, enc_y_prove];
if(new_wind_angle==90 || new_wind_angle==270)%This direction of the
wind will not deviate the vortex to the sides
    t_enc_prove(1) = norm(enc_prove-RPAS_pos)/norm(RPAS_spe)*3600;
end
%If the direction of the wind is in the same range of angles that the
%heading of the RPAS the point where the RPAS will intersect with the
%vortex will be after the intersection of the trajectories
if((0<=new_wind_angle)&& (new_wind_angle<90) || (270<new_wind_angle)&&
(new_wind_angle<=360))
    if((0<=new_RPAS_hea)&& (new_RPAS_hea<90) || (270<new_RPAS_hea)&&
(new_RPAS_hea<=360))
        t_enc_prove(1) = norm(enc_prove-RPAS_pos)/norm(RPAS_spe)*3600;
    else%if not the encounter will be before the intersection point
        t_enc_prove(1) = -norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;
    end
else
    if((0<=new_RPAS_hea)&& (new_RPAS_hea<90) || (270<new_RPAS_hea)&&
(new_RPAS_hea<=360))
        t_enc_prove(1) = -norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;
    else
        t_enc_prove(1) = norm(enc_prove-RPAS_pos)/norm(RPAS_spe)*3600;
    end
end
%Find time to encounter
encontrado=false;
conflict=true;
i=2;
iteration=0;
while(encontrado==false)
airliner_pos(i) = airliner_pos(1) + airliner_spe * t_enc_prove(i-
1)/3600;

enc_y_prove = airliner_pos(i) * tand(alpha);
enc_x_prove = enc_y_prove/tand(beta);
enc_prove = [enc_x_prove, enc_y_prove];
if(new_wind_angle==90 || new_wind_angle==270)
    t_enc_prove(i) = norm(enc_prove-RPAS_pos)/norm(RPAS_spe)*3600;
end
if((0<=new_wind_angle)&& (new_wind_angle<90) || (270<new_wind_angle)&&
(new_wind_angle<=360))
    if((0<=new_RPAS_hea)&& (new_RPAS_hea<90) || (270<new_RPAS_hea)&&
(new_RPAS_hea<=360))
        t_enc_prove(i) = norm(enc_prove-RPAS_pos)/norm(RPAS_spe)*3600;
    else
        t_enc_prove(i) = -norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;
    end
else
```

```
    if((0<=new_RPAS_hea)&& (new_RPAS_hea<90) || (270<new_RPAS_hea)&&
(new_RPAS_hea<=360))
        t_enc_prove(i) = -norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;
    else
        t_enc_prove(i) = norm(enc_prove-RPAS_pos)/norm(RPAS_spe)*3600;
    end
end
t_diff=abs(t_enc_prove(i)-t_enc_prove(i-1));
if(t_diff<=0.01)
    encontrado=true;
    enc_y=enc_y_prove;
    enc_x=enc_x_prove;
    t_enc=t_enc_prove(i);

end

iteration=iteration+1;
if(iteration==300)
    encontrado=true;
    t_enc=2000;
    conflict=false;
end
i=i+1;
end
if(conflict==false)
    disp('The RPAS never intersects the trajectory of the vortex');
else

    % Vertical analysis

    %Distance between the encounter point and where is the airliner
    d_vortex=abs(((airliner_pos(i-1)-enc_x)^2+(0-enc_y)^2)^(1/2));
    %With this distance we can compute the time when the vortex was
generated
    %from the encounter point.
    t_vortex_gen=(d_vortex/airliner_spe)*60;%minutes
    %parameters needed to comute the downwash velocity
    h_airliner_m=h_airliner/3;% airliner altitude in meters
    T=15-6.5*(h_airliner_m/1000);% temperature at the airliner
altitude in degrees
    T=T+273;% temperature in kelvins
    den=1.225*exp(-(9.81/(287*T))*h_airliner_m);%density at airliner
altitude (kg/m3)
    V=airliner_spe*(1852/3600);%airliner speed in m/s
    circ=(9.81*airliner_mass)/(den*V*0.8*airliner_span);%(m^2/s)
    downwash=circ/(2*pi*0.8*airliner_span);%vortex downwash velocity
in m/s
    downwash=downwash*3*60;%vortex downwash velocity in ft/min

    %Time the vortex needs to stabilize at 1000ft
    if(vert_wind_V==0)

        t_vortex_stable=1000/downwash;%minutes
        if(t_vortex_gen > t_vortex_stable)
            h_vortex=h_airliner-1000;%ft
            vertical_limit_sup=h_vortex + 65;%ft
            vertical_limit_inf=h_vortex - 65;%ft
        else
```

```
            %It is computed how many feet is the vortex below the
airliner
            h_vortex=h_airliner-500*t_vortex_gen;
            vertical_limit_sup=h_vortex + 65; %ft
            vertical_limit_inf=h_vortex - 65;%ft

        end
    else
      decay_rate=downwash + vert_wind_V;%ft/min
      t_vortex_stable=1000/decay_rate;%minutes

        if(t_vortex_gen > t_vortex_stable)
            h_vortex=h_airliner-1000;%ft
            vertical_limit_sup=h_vortex + 65;%ft
            vertical_limit_inf=h_vortex - 65;%ft
        else
            %It is computed how many feet is the vortex below the
airliner
            h_vortex=h_airliner-decay_rate*t_vortex_gen;
            vertical_limit_sup=h_vortex + 65; %ft
            vertical_limit_inf=h_vortex - 65;%ft

        end
    end
    if(vertical_limit_inf> h_RPAS || h_RPAS > h_airliner ||
vertical_limit_sup < h_RPAS)
        disp('The RPAS is not inside the vortex vertical hazard area.
There is no conflict');
    else
        % Horizontal analysis
    % Computation of the distance that the RPAS has flown
    x_RPAS_enc=RPAS_pos(1)+RPAS_spe(1)*(t_enc/3600);
    y_RPAS_enc=RPAS_pos(2)+RPAS_spe(2)*(t_enc/3600);
    d_RPAS=(x_RPAS_enc^2+y_RPAS_enc^2)^(1/2);

    %Computation of the distance the airliner has flown
    position_airliner=airliner_pos(1)+airliner_spe*(t_enc/3600);
    d_airliner=position_airliner-airliner_pos(1);

    %Knowing the time the vortex stays, it can be computed if the RPAS
is
    %inside the vortex or not
    t_vortex=5*60;%it  stays 5 min as maximum
    %with this time it is computed the distance the aircraft has flown
and that
    %distance will be where the vortex will stay
    horizontal_vortex_area=airliner_spe*(t_vortex/3600);
    % It is computed the distance between the RPAS and the airliner
    RPAS_airliner_dist=abs(((position_airliner-x_RPAS_enc)^2+(0-
y_RPAS_enc)^2)^(1/2));
    %Both distances are compared, if the distance between the RPAS and
the
    %airliner is smaller, the RPAS is inside the vortex area. If it is
larger
    %it will be outside the area.
    if(horizontal_vortex_area>RPAS_airliner_dist)
        disp('The RPAS is inside the vertical and horizontal vortex
area')
        %Intensity
        t_vortex_gen=t_vortex_gen*60;%seconds
```

```
        F=1-
2*(2*0.035*(airliner_span/RPAS_span))*((1+(2*0.035*(airliner_span/RPAS
_span))^2)^(1/2)-(2*0.035*(airliner_span/RPAS_span)));
        if(atmosphere1==true)
           fitresult = createFit(t1, c1);
           coeffval=coeffvalues(fitresult);
           total_circ=coeffval(1)*t_vortex_gen.^9 +
coeffval(2)*t_vortex_gen.^8 + coeffval(3)*t_vortex_gen.^7 +
coeffval(4)*t_vortex_gen.^6 + coeffval(5)*t_vortex_gen.^5 +
coeffval(6)*t_vortex_gen.^4 + coeffval(7)*t_vortex_gen.^3 +
coeffval(8)*t_vortex_gen.^2 +  coeffval(9)*t_vortex_gen +
coeffval(10);
        end
        if(atmosphere2==true)
           fitresult = createFit(t2, c2);
           coeffval=coeffvalues(fitresult);
           total_circ=coeffval(1)*t_vortex_gen.^9 +
coeffval(2)*t_vortex_gen.^8 + coeffval(3)*t_vortex_gen.^7 +
coeffval(4)*t_vortex_gen.^6 + coeffval(5)*t_vortex_gen.^5 +
coeffval(6)*t_vortex_gen.^4 + coeffval(7)*t_vortex_gen.^3 +
coeffval(8)*t_vortex_gen.^2 +  coeffval(9)*t_vortex_gen +
coeffval(10);
        end
        if(atmosphere3==true)
            fitresult = createFit(t3, c3);
            coeffval=coeffvalues(fitresult);
            total_circ= coeffval(1)*t_vortex_gen.^7 +
coeffval(2)*t_vortex_gen.^6 + coeffval(3)*t_vortex_gen.^5 +
coeffval(4)*t_vortex_gen.^4 + coeffval(5)*t_vortex_gen.^3 +
coeffval(6)*t_vortex_gen.^2 + coeffval(7)*t_vortex_gen + coeffval(8);
        end
        total_circ=total_circ-575;
        total_circ=circ+total_circ;


rmc=(total_circ/(RPAS_speed*(1852/3600)*RPAS_span))*(AR_RPAS/(AR_RPAS+
4))*F;%Rolling Moment Coefficient created by the vortex
        if(global_hawk==true)
            lambda=1/3;
            Cl_alpha=0.105*180/pi;
        else if(ikhana==true)
                lambda=0.384;
                Cl_alpha=0.122*180/pi;
            end
        end
        Cl_p=-(Cl_alpha*(1+3*lambda))/(12*(1+lambda));
        Cl_control=-Cl_p*0.07;
        normalized=rmc/Cl_control;

        if(normalized > 1)
            disp('The RPAS cannot perform this trajectory because the
wake vortex will turn it upside down');
        else
            disp('Safe situation, the wake vortex will not be able to
turn the RPAS upside down')
        end

    else
        disp('The RPAS is outside the vortex area, there is no risk.
');
    end
```

```
    end
end

end
```

Function: createFit

```
function [fitresult, gof] = createFit(t1, c1)
%CREATEFIT(T1,C1)
%  Create a fit.
%
%  Data for 'untitled fit 1' fit:
%      X Input : t1
%      Y Output: c1
%  Output:
%      fitresult : a fit object representing the fit.
%      gof : structure with goodness-of fit info.
%
%  See also FIT, CFIT, SFIT.

%  Auto-generated by MATLAB on 30-May-2016 13:24:31


%% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( t1, c1 );

% Set up fittype and options.
ft = fittype( 'poly9' );

% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );

% Plot fit with data.
% figure( 'Name', 'untitled fit 1' );
% h = plot( fitresult, xData, yData );
% legend( h, 'c1 vs. t1', 'untitled fit 1', 'Location', 'NorthEast' );
% Label axes
% xlabel t1
% ylabel c1
% grid on
```

Appendix B. Matlab code. Collision model with turns

Main

```
close all;
d=3;%NM
i=1;
phy=0;
turns=true;
b=20;%(°)
RPAS_change_hdg=30;%(°)
t1=[0 9 18 27 39 57 72 78 84 93 102 105 108 111 114 117 120 123 126
129 138 147 156 165];
c1=[575 550 540 520 500 480 460 450 440 430 420 410 400 390 360 340
320 300 280 260 200 150 100 70];
t2=[0 3 9 12 21 24 27 30 33 39 45 48 57 63 75 84 96 105];
c2=[575 560 550 540 530 520 510 500 480 440 400 360 300 260 200 160
100 70];
t3=[0 9 12 21 30 39 51 57 66 72 78 81 84 87 90 93 96 102 105 114 120];
c3=[575 550 540 530 510 500 490 480 470 460 440 420 400 380 320 300
260 220 190 100 70];
atmosphere1=true;
atmosphere2=false;
atmosphere3=false;
global_hawk=true;
ikhana=false;
m=1;
while(phy<360)
    angle(i)=phy;
    hdg_RPAS=phy;
    t=1;
    %Airliner data is introduced
    airliner_span=35.8;% airliner wingspan in meters
    airliner_mass=64500; %airliner mass in kg (Max landing weight
A320)
    X0_airliner=5;%Initial x position (NM)
    Y0_airliner=5;%Initial y position (NM)
    V_airliner=500;%Airliner speed (knots)
    h_airliner=35000;%Airliner altitud (ft)
    hdg_airliner=180;%Airliner heading (°)
    %Wind data is introduced
    beta=90;%Direction to where the wind is blowing (°)
    w=100;%Wind speed(knots)
    vert_wind_V=0; %Vertical wind speed (wind speed in the z
direction) (ft/min). Positive if it goes downwards and negative if it
goes upwards.
    % The x and y speed components are computed
    u_airliner=V_airliner*sind(hdg_airliner);%Airliner velocity in the
x direction (knots)
    v_airliner=V_airliner*cosd(hdg_airliner);%Airliner velocity in the
y direction (knots)
    %A new point is created to make the airliner line equation
    X1_airliner=X0_airliner+u_airliner*t;
    Y1_airliner=Y0_airliner+v_airliner*t;
    %Vector of the line is created
    vect_1_airliner=X1_airliner-X0_airliner;
    vect_2_airliner=Y1_airliner-Y0_airliner;
    %Airliner equation line
    %Y_airliner=((X_airliner-
X0_airliner)/vect_1_airliner)*vect_2_airliner+Y0_airliner;
```

```
    if(hdg_RPAS==hdg_airliner||abs(hdg_airliner-hdg_RPAS)==180 )
        hdg_RPAS=hdg_RPAS+1;
    end
    %RPAS data is introduced
    if(global_hawk==true)
        AR_RPAS=25;
        RPAS_span=39.9;%RPAS wingspan in meters (global hawk)
    else if(ikhana==true)
            AR_RPAS=17;
            RPAS_span=20.1;%RPAS wingspan in meters (Ikhana)
        end
    end
    X0_RPAS=-d*sind(hdg_RPAS)+X0_airliner;%Initial RPAS x position
(NM)
    Y0_RPAS=-d*cosd(hdg_RPAS)+Y0_airliner;%Initial RPAS y position
(NM)
    V_RPAS=300;%RPAS speed (knots)
    h_RPAS=34500;%RPAS altitude (ft)


    %The velocity x and y components are computed
    u_RPAS=V_RPAS*sind(hdg_RPAS);% RPAS velocity in the x direction
(knots)
    v_RPAS=V_RPAS*cosd(hdg_RPAS);%RPAS velocity in the y direction
(knots)
    %A new point is created to make the RPAS line equation
    X1_RPAS=X0_RPAS+u_RPAS*t;
    Y1_RPAS=Y0_RPAS+v_RPAS*t;
    %Vector of the line is created
    vect_1_RPAS=X1_RPAS-X0_RPAS;
    vect_2_RPAS=Y1_RPAS-Y0_RPAS;
    %RPAS line equation
    %Y_uav=((X_uav-Xo_uav)/vect_1_uav)*vect_2_uav+Y0_uav;

    %If the RPAS and the airliner fly at the same altitude  with
opposite headings, they will have a front collision
    if(abs(hdg_airliner-hdg_RPAS)==180 && h_RPAS==h_airliner)
        disp('A frontal collision will be produced')
    else
        %The two line equations are equated and the x intersection is
        %obtained
        X_intersec=(-
X0_airliner*vect_2_airliner*vect_1_RPAS+vect_1_airliner*Y0_airliner*ve
ct_1_RPAS+vect_1_airliner*X0_RPAS*vect_2_RPAS-
vect_1_airliner*vect_1_RPAS*Y0_RPAS)/(vect_1_airliner*vect_2_RPAS-
vect_2_airliner*vect_1_RPAS);

        %Substituing the x value of the intersection to one of the two
equations, the y value of the
        %intersection is obtained
        Y_intersec=((X_intersec-
X0_RPAS)/vect_1_RPAS)*vect_2_RPAS+Y0_RPAS;
        %To be able to compute the time the RPAS needs to get the
        %intersection point, first it must known the distance that
there
        %is. As the initial point where the RPAS and the intersection
point
        %are known the distance can be calculated.
```

```
        d_RPAS_intersec=((X_intersec-X0_RPAS)^2+(Y_intersec-
Y0_RPAS)^2)^(1/2);
        %Knowing the distance travelled and the speed at which it
flies, the time can be known
        t_RPAS_intersec=d_RPAS_intersec/V_RPAS;

        %To know the time the airliner needs to get to the
intersection point,
        %first the distance it will travel until this point is
computed
        d_airliner_intersec=((X_intersec-X0_airliner)^2+(Y_intersec-
Y0_airliner)^2)^(1/2);
        %With this distance and the speed at which it flies, the time
the airliner needs to get
        %to the intersection point is calculated
        t_airliner_intersec=d_airliner_intersec/V_airliner;

        %The time the airliner travels after the intersection point
        t_airliner_after_inter=t_RPAS_intersec-t_airliner_intersec;

        %It is analysed if the RPAS is before or after the airliner at
the
        %intersection point
        if(t_airliner_after_inter<0)
            disp('There is no conflict, the RPAS arrive to the
intersection point before the airliner');
        else

[t_enc(i),normalized]=detectConflictWind(ikhana,global_hawk,AR_RPAS,at
mosphere1,atmosphere2,atmosphere3,RPAS_span,airliner_mass,airliner_spa
n,RPAS_change_hdg,b,turns,t_airliner_after_inter,vert_wind_V,h_airline
r,h_RPAS,hdg_airliner,hdg_RPAS,beta,X0_airliner,Y0_airliner,V_airliner
,X_intersec,Y_intersec,w,V_RPAS,t1,c1,t2,c2,t3,c3);
            if(normalized~=0)
                    vect_norm(m)=normalized;
                    vect_angle(m)=hdg_RPAS;
                    m=m+1;
            end


        end
    end
    i=i+1;
    phy=phy+1;

end
figure(1)
plot(angle,t_enc);
hold on;
title('Heading variation = 20°');
xlabel('RPAS heading (°)');
ylabel('Time to encounter the vortex from the intersection point(s)');

figure(2)
plot(vect_angle,vect_norm);
hold on
title('Normalized Cl vs RPAS heading (Global Hawk)');
xlabel('RPAS heading (°)');
ylabel('Cl/Cl control');
```

Function: DetectConflictWind

```
function[t_enc,normalized]=detectConflictWind(ikhana,global_hawk,AR_RP
AS,atmosphere1,atmosphere2,atmosphere3,RPAS_span,airliner_mass,airline
r_span,RPAS_change_hdg,b,turns,t_airliner_after_inter,vert_wind_V,h_ai
rliner,h_RPAS,hdg_airliner,hdg_RPAS,beta,X0_airliner,Y0_airliner,V_air
liner,X_intersec,Y_intersec,w,V_RPAS,t1,c1,t2,c2,t3,c3)
%%Initialize variables
%Intersection of trajectories
X_intersec=0;
Y_intersec=0;
%RPAS position
RPAS_pos = [0,0];
%RPAS velocity
RPAS_speed=V_RPAS;
%Airliner position
airliner_pos_x=0;
airliner_pos_y=0;
%airliner heading
airliner_hea = hdg_airliner;
%Airliner speed
airliner_spe=V_airliner;% (knots)
%Wind
wind_speed=w;%(knots)
%wind angle
wind_angle=beta;%(°)
%RPAS heading
RPAS_hea=hdg_RPAS;%(°)
normalized=0;

enc_x=0;
enc_y=0;

%the airliners always flies to the east, so the airliner and the RPAS
are
%rotated in order to make the airliner fly with a heading of 90
degrees.
angle_rotation=90-airliner_hea;
airliner_hea2=airliner_hea+angle_rotation;%It must be 90
RPAS_hea2=RPAS_hea+ angle_rotation;
if(RPAS_hea2<0)
    RPAS_hea2=360+RPAS_hea2;
end
wind_angle2=wind_angle+angle_rotation;
if(wind_angle2<0)
    wind_angle2=360+wind_angle2;
end
new_RPAS_hea=RPAS_hea2;
new_wind_angle=wind_angle2;
new_airliner_hea=airliner_hea2;
wind_spe = [wind_speed * sind(new_wind_angle), wind_speed *
cosd(new_wind_angle)];
RPAS_spe = [RPAS_speed * sind(new_RPAS_hea), RPAS_speed *
cosd(new_RPAS_hea)];

%At t=0 the RPAS is in the intersection point. Then, situate airliner
%following the initial conditions
if(turns==true)
    %Position where the airliner will be after the intersection point
if it
```

```
    %performs a turn
    [X_airliner_RPAS_intersec, Y_airliner_RPAS_intersec,
airliner_hea]=turn(RPAS_change_hdg,b,X_intersec,Y_intersec,airliner_sp
e,t_airliner_after_inter,airliner_hea2);

    %The airliner will be now at the origin
    airliner_pos_x= X_airliner_RPAS_intersec-X_airliner_RPAS_intersec;
    airliner_pos_y=Y_airliner_RPAS_intersec-Y_airliner_RPAS_intersec;
    %The new RPAS position is computed, based that the airliner is at
the
    %origin
    RPAS_pos=[0-X_airliner_RPAS_intersec,0-Y_airliner_RPAS_intersec];
    %Next step is assigning to the airliner a heading of 90° in order
to be
    %able to use the algorithm performed to find the encounter
position
    angle_rotation=90-airliner_hea;
    new_airliner_hea=airliner_hea+angle_rotation;%It must be 90
    new_RPAS_hea=RPAS_hea2+ angle_rotation;
    if(new_RPAS_hea<0)
        new_RPAS_hea=360+new_RPAS_hea;
    end
    if(new_RPAS_hea==0 || new_RPAS_hea==360)
        new_RPAS_hea=1;
    end
    if(new_RPAS_hea==180)
        new_RPAS_hea=181;
    end
    if(new_RPAS_hea==270)
        new_RPAS_hea=271;
    end
    if(new_RPAS_hea==90)
        new_RPAS_hea=91;
    end
    new_wind_angle=wind_angle2+angle_rotation;
    if(new_wind_angle<0)
        new_wind_angle=360+new_wind_angle;
    end
    %The new RPAS position  after the rotation

RPAS_pos=[RPAS_pos(1)*cosd(angle_rotation)+RPAS_pos(2)*sind(angle_rota
tion), -
RPAS_pos(1)*sind(angle_rotation)+RPAS_pos(2)*cosd(angle_rotation)];
    wind_spe = [wind_speed * sind(new_wind_angle), wind_speed *
cosd(new_wind_angle)];
    RPAS_spe = [RPAS_speed * sind(new_RPAS_hea), RPAS_speed *
cosd(new_RPAS_hea)];
    %calculate vortex angle
    alpha = atand(wind_spe(2)/(wind_spe(1) + airliner_spe(1)));
    %Definition of the vortex line
    X1_vortex=-1;
    Y1_vortex=tand(alpha);
    X0_vortex=airliner_pos_x;
    Y0_vortex=airliner_pos_y;
    vect_1_vortex=X1_vortex-X0_vortex;
    vect_2_vortex=Y1_vortex-Y0_vortex;
    %Definition of the RPAS trajectory
    X0_RPAS=RPAS_pos(1);
    Y0_RPAS=RPAS_pos(2);
    X1_RPAS=X0_RPAS+RPAS_speed*sind(new_RPAS_hea)*1;
    Y1_RPAS=Y0_RPAS+RPAS_speed*cosd(new_RPAS_hea)+1;
```

```
    vect_1_RPAS=X1_RPAS-X0_RPAS;
    vect_2_RPAS=Y1_RPAS-Y0_RPAS;
    %The two line equations are equated and the x intersection is
    %obtained
    enc_x_prove=(-
X0_vortex*vect_2_vortex*vect_1_RPAS+vect_1_vortex*Y0_vortex*vect_1_RPA
S+vect_1_vortex*X0_RPAS*vect_2_RPAS-
vect_1_vortex*vect_1_RPAS*Y0_RPAS)/(vect_1_vortex*vect_2_RPAS-
vect_2_vortex*vect_1_RPAS);
    %Substituing the x value of the intersection to one of the two
equations, the y value of the
    %intersection is obtained
    enc_y_prove=((enc_x_prove-
X0_RPAS)/vect_1_RPAS)*vect_2_RPAS+Y0_RPAS;
    %Vortex encounter position
    enc_prove = [enc_x_prove, enc_y_prove];

    if((0<=new_RPAS_hea)&& (new_RPAS_hea<90) || (270<new_RPAS_hea)&&
(new_RPAS_hea<=360))
        if(RPAS_pos(2)<=enc_y_prove)
            t_enc_prove(1) = norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;%seconds
        else
            t_enc_prove(1) = -norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;%seconds
        end
    else
        if(RPAS_pos(2)< enc_y_prove)
            t_enc_prove(1) = -norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;%seconds
        else
            t_enc_prove(1) = norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;%seconds
        end
    end

    if(new_RPAS_hea==90)
        if(RPAS_pos(1)> enc_x_prove)
            t_enc_prove(1) = -norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;%seconds
        else
            t_enc_prove(1) = norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;%seconds
        end
    elseif(new_RPAS_hea==270)
        if(RPAS_pos(1)>=enc_x_prove)
            t_enc_prove(1) = norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;%seconds
        else
            t_enc_prove(1) = -norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;%seconds
        end
    end


else

    airliner_pos_x(1) = airliner_pos_x +
airliner_spe*t_airliner_after_inter;
    %calculate vortex angle
```

```
    alpha = atand(wind_spe(2)/(wind_spe(1) + airliner_spe(1)));
    %calculate angle of the conflict
    beta = 90 - new_RPAS_hea;
    %find encounter angle
    gamma = 180 - beta - alpha;
    %find vortex encounter position
    enc_y_prove= airliner_pos_x * tand(alpha);
    enc_x_prove= enc_y_prove/tand(beta);
    enc_prove = [enc_x_prove, enc_y_prove];
    if(new_wind_angle==90 || new_wind_angle==270)%This direction of
the wind will not deviate the vortex to the sides
    t_enc_prove(1) = norm(enc_prove-RPAS_pos)/norm(RPAS_spe)*3600;
    end
    %If the direction of the wind is in the same range of angles that
the
    %heading of the RPAS the point where the RPAS will intersect with
the
    %vortex will be after the intersection of the trajectories
    if((0<=new_wind_angle)&& (new_wind_angle<90) ||
(270<new_wind_angle)&& (new_wind_angle<=360))
        if((0<=new_RPAS_hea)&& (new_RPAS_hea<90) ||
(270<new_RPAS_hea)&& (new_RPAS_hea<=360))
            t_enc_prove(1) = norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;%seconds
        else%if not the encounter will be before the intersection
point
            t_enc_prove(1) = -norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;
        end
    else
        if((0<=new_RPAS_hea)&& (new_RPAS_hea<90) ||
(270<new_RPAS_hea)&& (new_RPAS_hea<=360))
            t_enc_prove(1) = -norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;
        else
            t_enc_prove(1) = norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;
        end
    end
end



%Find time to encounter
encontrado=false;
conflict=true;
i=2;
iteration=0;
while(encontrado==false)

    if (turns==true)

        t=t_airliner_after_inter+(t_enc_prove(i-1)/3600);
        if(t>=0)

            [X_airliner_RPAS_intersec, Y_airliner_RPAS_intersec,
airliner_hea]=turn(RPAS_change_hdg,b,X_intersec,Y_intersec,airliner_sp
e,t,airliner_hea2);

            %The airliner will be now at the origin
```

```
            airliner_pos_x(i)= X_airliner_RPAS_intersec-
X_airliner_RPAS_intersec;
            airliner_pos_y=Y_airliner_RPAS_intersec-
Y_airliner_RPAS_intersec;
            %The new RPAS position is computed, based that the
airliner is at the
            %origin
            RPAS_pos=[0-X_airliner_RPAS_intersec,0-
Y_airliner_RPAS_intersec];
            %Next step is assigning to the airliner a heading of 90°
in order to be
            %able to use the algorithm performed to find the encounter
position
            angle_rotation=90-airliner_hea;
            new_airliner_hea=airliner_hea+angle_rotation;%It must be
90
            new_RPAS_hea=RPAS_hea2+ angle_rotation;
            if(new_RPAS_hea<0)
                new_RPAS_hea=360+new_RPAS_hea;
            end
            if(new_RPAS_hea==0 || new_RPAS_hea==360)
                new_RPAS_hea=1;
            end
            if(new_RPAS_hea==180)
                new_RPAS_hea=181;
            end
            if(new_RPAS_hea==270)
                new_RPAS_hea=271;
            end
            if(new_RPAS_hea==90)
                new_RPAS_hea=91;
            end
            new_wind_angle=wind_angle2+angle_rotation;
            if(new_wind_angle<0)
                new_wind_angle=360+new_wind_angle;
            end
            %The new RPAS position  after the rotation

RPAS_pos=[RPAS_pos(1)*cosd(angle_rotation)+RPAS_pos(2)*sind(angle_rota
tion), -
RPAS_pos(1)*sind(angle_rotation)+RPAS_pos(2)*cosd(angle_rotation)];
            wind_spe = [wind_speed * sind(new_wind_angle), wind_speed
* cosd(new_wind_angle)];
            RPAS_spe = [RPAS_speed * sind(new_RPAS_hea), RPAS_speed *
cosd(new_RPAS_hea)];

            %calculate vortex angle
            alpha = atand(wind_spe(2)/(wind_spe(1) +
airliner_spe(1)));
            %Definition of the vortex line
            X1_vortex=-1;
            Y1_vortex=tand(alpha);
            X0_vortex=airliner_pos_x;
            Y0_vortex=airliner_pos_y;
            vect_1_vortex=X1_vortex-X0_vortex;
            vect_2_vortex=Y1_vortex-Y0_vortex;
            %Definition of the RPAS trajectory
            X0_RPAS=RPAS_pos(1);
            Y0_RPAS=RPAS_pos(2);
            X1_RPAS=X0_RPAS+RPAS_speed*sind(new_RPAS_hea);
            Y1_RPAS=Y0_RPAS+RPAS_speed*cosd(new_RPAS_hea);
```

```
            vect_1_RPAS=X1_RPAS-X0_RPAS;
            vect_2_RPAS=Y1_RPAS-Y0_RPAS;
            %The two line equations are equated and the x intersection
is
            %obtained
            enc_x_prove=(-
X0_vortex*vect_2_vortex*vect_1_RPAS+vect_1_vortex*Y0_vortex*vect_1_RPA
S+vect_1_vortex*X0_RPAS*vect_2_RPAS-
vect_1_vortex*vect_1_RPAS*Y0_RPAS)/(vect_1_vortex*vect_2_RPAS-
vect_2_vortex*vect_1_RPAS);
            %Substituing the x value of the intersection to one of the
two equations, the y value of the
            %intersection is obtained
            enc_y_prove=((enc_x_prove-
X0_RPAS)/vect_1_RPAS)*vect_2_RPAS+Y0_RPAS;
            %Vortex encounter position
            enc_prove = [enc_x_prove, enc_y_prove];

            if((0<=new_RPAS_hea)&& (new_RPAS_hea<90) ||
(270<new_RPAS_hea)&& (new_RPAS_hea<=360))
                if(RPAS_pos(2)<=enc_y_prove)
                    t_enc_prove(i) = norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;%seconds
                else
                    t_enc_prove(i) = -norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;%seconds
                end
            else
                if(RPAS_pos(2)< enc_y_prove)
                    t_enc_prove(i) = -norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;%seconds
                else
                    t_enc_prove(i) = norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;%seconds
                end
            end

            if(new_RPAS_hea==90)
                if(RPAS_pos(1)> enc_x_prove)
                    t_enc_prove(i) = -norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;%seconds
                else
                    t_enc_prove(i) = norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;%seconds
                end
            elseif(new_RPAS_hea==270)
                if(RPAS_pos(1)>=enc_x_prove)
                    t_enc_prove(i) = norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;%seconds
                else
                    t_enc_prove(i) = -norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;%seconds
                end
            end
            t_diff=abs(t_enc_prove(i)-t_enc_prove(i-1));
            if(t_diff<=0.01)
                encontrado=true;
                enc_y=enc_y_prove;
                enc_x=enc_x_prove;
                t_enc=t_enc_prove(i);
```

```
            end

        else
            encontrado=true;
            conflict=false;
            t_enc=2000;
        end
    else
        airliner_pos_x(i) = airliner_pos_x(1) + airliner_spe *
t_enc_prove(i-1)/3600;
        enc_y_prove = airliner_pos_x(i) * tand(alpha);
        enc_x_prove = enc_y_prove/tand(beta);
        enc_prove = [enc_x_prove, enc_y_prove];

        if(new_wind_angle==90 || new_wind_angle==270)
            t_enc_prove(i) = norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;
        end
        if((0<=new_wind_angle)&& (new_wind_angle<90) ||
(270<new_wind_angle)&& (new_wind_angle<=360))
            if((0<=new_RPAS_hea)&& (new_RPAS_hea<90) ||
(270<new_RPAS_hea)&& (new_RPAS_hea<=360))
                t_enc_prove(i) = norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;
            else
                t_enc_prove(i) = -norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;
            end
        else
            if((0<=new_RPAS_hea)&& (new_RPAS_hea<90) ||
(270<new_RPAS_hea)&& (new_RPAS_hea<=360))
                t_enc_prove(i) = -norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;
            else
                t_enc_prove(i) = norm(enc_prove-
RPAS_pos)/norm(RPAS_spe)*3600;
            end
        end
        t_diff=abs(t_enc_prove(i)-t_enc_prove(i-1));
        if(t_diff<=0.01)
            encontrado=true;
            enc_y=enc_y_prove;
            enc_x=enc_x_prove;
            t_enc=t_enc_prove(i);

        end
    end



    iteration=iteration+1;
    if(iteration==300)
        encontrado=true;
        t_enc=2000;
        conflict=false;
    end
    i=i+1;
end
if(conflict==false)
    disp('The RPAS never intersects the trajectory of the vortex');
```

```
else

    % Vertical analysis

    %Distance between the encounter point and where is the airliner
    d_vortex=abs(((airliner_pos_x(i-1)-enc_x)^2+(0-enc_y)^2)^(1/2));
    %With this distance we can compute the time when the vortex was
generated
    %from the encounter point.
    t_vortex_gen=(d_vortex/airliner_spe)*60;%minutes
    %parameters needed to comute the downwash velocity
    h_airliner_m=h_airliner/3;% airliner altitude in meters
    T=15-6.5*(h_airliner_m/1000);% temperature at the airliner
altitude in degrees
    T=T+273;% temperature in kelvins
    den=1.225*exp(-(9.81/(287*T))*h_airliner_m);%density at airliner
altitude (kg/m3)
    V=airliner_spe*(1852/3600);%airliner speed in m/s
    circ=(9.81*airliner_mass)/(den*V*0.8*airliner_span);
    downwash=circ/(2*pi*0.8*airliner_span);%vortex downwash velocity
in m/s
    downwash=downwash*3*60;%vortex downwash velocity in ft/min

    %Time the vortex needs to stabilize at 1000ft
    if(vert_wind_V==0)

        t_vortex_stable=1000/downwash;%minutes
        if(t_vortex_gen > t_vortex_stable)
            h_vortex=h_airliner-1000;%ft
            vertical_limit_sup=h_vortex + 65;%ft
            vertical_limit_inf=h_vortex - 65;%ft
        else
            %It is computed how many feet is the vortex below the
airliner
            h_vortex=h_airliner-500*t_vortex_gen;
            vertical_limit_sup=h_vortex + 65; %ft
            vertical_limit_inf=h_vortex - 65;%ft

        end
    else
        decay_rate=downwash + vert_wind_V;%ft/min
        t_vortex_stable=1000/decay_rate;%minutes

        if(t_vortex_gen > t_vortex_stable)
            h_vortex=h_airliner-1000;%ft
            vertical_limit_sup=h_vortex + 65;%ft
            vertical_limit_inf=h_vortex - 65;%ft
        else
            %It is computed how many feet is the vortex below the
airliner
            h_vortex=h_airliner-decay_rate*t_vortex_gen;
            vertical_limit_sup=h_vortex + 65; %ft
            vertical_limit_inf=h_vortex - 65;%ft

        end
    end
    if(vertical_limit_inf> h_RPAS || h_RPAS > h_airliner ||
vertical_limit_sup < h_RPAS)
        disp('The RPAS is not inside the vortex vertical hazard area.
There is no conflict');
```

```
    else
        % Horizontal analysis
        % Computation of the distance that the RPAS has flown
        x_RPAS_enc=RPAS_pos(1)+RPAS_spe(1)*(t_enc/3600);
        y_RPAS_enc=RPAS_pos(2)+RPAS_spe(2)*(t_enc/3600);
        d_RPAS=(x_RPAS_enc^2+y_RPAS_enc^2)^(1/2);

        %Computation of the distance the airliner has flown
        position_airliner=airliner_pos_x(1)+airliner_spe*(t_enc/3600);
        d_airliner=position_airliner-airliner_pos_x(1);

        %Knowing the time the vortex stays, it can be computed if the
RPAS is
        %inside the vortex or not
        t_vortex=5*60;%it  stays 5 min as maximum
        %with this time it is computed the distance the aircraft has
flown and that
        %distance will be where the vortex will stay
        horizontal_vortex_area=airliner_spe*(t_vortex/3600);
        % It is computed the distance between the RPAS and the
airliner
        RPAS_airliner_dist=abs(((position_airliner-x_RPAS_enc)^2+(0-
y_RPAS_enc)^2)^(1/2));
        %Both distances are compared, if the distance between the RPAS
and the
        %airliner is smaller, the RPAS is inside the vortex area. If
it is larger
        %it will be outside the area.
        if(horizontal_vortex_area>RPAS_airliner_dist)
            disp('The RPAS is inside the vortex ')
            %Intensity
            t_vortex_gen=t_vortex_gen*60;%seconds
            F=1-
2*(2*0.035*(airliner_span/RPAS_span))*((1+(2*0.035*(airliner_span/RPAS
_span))^2)^(1/2)-(2*0.035*(airliner_span/RPAS_span)));
            if(atmosphere1==true)
               fitresult = createFit(t1, c1);
               coeffval=coeffvalues(fitresult);
               total_circ=coeffval(1)*t_vortex_gen.^9 +
coeffval(2)*t_vortex_gen.^8 + coeffval(3)*t_vortex_gen.^7 +
coeffval(4)*t_vortex_gen.^6 + coeffval(5)*t_vortex_gen.^5 +
coeffval(6)*t_vortex_gen.^4 + coeffval(7)*t_vortex_gen.^3 +
coeffval(8)*t_vortex_gen.^2 +  coeffval(9)*t_vortex_gen +
coeffval(10);
            end
            if(atmosphere2==true)
               fitresult = createFit(t2, c2);
               coeffval=coeffvalues(fitresult);
               total_circ=coeffval(1)*t_vortex_gen.^9 +
coeffval(2)*t_vortex_gen.^8 + coeffval(3)*t_vortex_gen.^7 +
coeffval(4)*t_vortex_gen.^6 + coeffval(5)*t_vortex_gen.^5 +
coeffval(6)*t_vortex_gen.^4 + coeffval(7)*t_vortex_gen.^3 +
coeffval(8)*t_vortex_gen.^2 +  coeffval(9)*t_vortex_gen +
coeffval(10);
            end
            if(atmosphere3==true)
                fitresult = createFit(t3, c3);
                coeffval=coeffvalues(fitresult);
                total_circ= coeffval(1)*t_vortex_gen.^7 +
coeffval(2)*t_vortex_gen.^6 + coeffval(3)*t_vortex_gen.^5 +
```

```
coeffval(4)*t_vortex_gen.^4 + coeffval(5)*t_vortex_gen.^3 +
coeffval(6)*t_vortex_gen.^2 + coeffval(7)*t_vortex_gen + coeffval(8);
            end
            total_circ=total_circ-575;
            total_circ=circ+total_circ;


rmc=(total_circ/(RPAS_speed*(1852/3600)*RPAS_span))*(AR_RPAS/(AR_RPAS+
4))*F;%Rolling Moment Coefficient created by the vortex
            if(global_hawk==true)
                lambda=1/3;
                Cl_alpha=0.105*180/pi;
            else if(ikhana==true)
                    lambda=0.384;
                    Cl_alpha=0.122*180/pi;
                end
            end
            Cl_p=-(Cl_alpha*(1+3*lambda))/(12*(1+lambda));
            Cl_control=-Cl_p*0.07;
            normalized=rmc/Cl_control;
            if(normalized > 1)
                disp('The RPAS cannot perform this trajectory because
the wake vortex will turn it upside down');
            else
                disp('Safe situation, the wake vortex will not be able
to turn the RPAS upside down')
            end
        else
            disp('The RPAS is outside the vortex area, there is no
risk. ');
        end
    end
end

end
```

Funtion: Turn

```
function[X_airliner,Y_airliner,airliner_hea]=turn(RPAS_change_hdg,b,X_
initial,Y_initial,V_airliner,t,hdg_airliner)
g=9.81;%gravity m/s^2
max_delta_turn=RPAS_change_hdg*pi/180;%rad
turn_angle=(3*pi)/2;%initial turning angle (rad)
R=(V_airliner*1852/3600)^2/(g*tand(b)); %turning radius (m)
R=R/1852;%NM
l_max=max_delta_turn*R;%Maximum distance the airliner will travel in
the turn
t_turn=l_max/V_airliner;%Total time the airliner will need to make the
turn
l=V_airliner*t;%Distance the airliner will travel
if(t_turn>t)%The maneuver will be end in the turn
    delta_turn=l/R;%incremrent to apply to the the turning angle (rad)
    turn_angle=turn_angle+delta_turn;
    %New airliner position after the turn
    X_airliner=X_initial+R*cos(turn_angle);%Position x where the
airliner will be after the time given
    Y_airliner=Y_initial+R+R*sin(turn_angle);%Position y where the
airliner will be after the time given
```

```
    airliner_hea=hdg_airliner-(delta_turn*180/pi);%New airliner
heading after the maneuver
else
    delta_turn=max_delta_turn;
    turn_angle=turn_angle+delta_turn;
    X_airliner=X_initial+R*cos(turn_angle);%Position x of the airliner
when the turn to get the new heading ends
    Y_airliner=Y_initial+R+R*sin(turn_angle);%Position y of the
airliner when the turn to get the new heading ends
    l=l-l_max;% Distance left to travel after the turn
    airliner_hea=hdg_airliner-(delta_turn*180/pi); %New airliner
heading
    X_airliner=X_airliner+l*sind(airliner_hea);%Final x position of
the airliner
    Y_airliner=Y_airliner+l*cosd(airliner_hea);%Final y position of
the airliner
end

end
```